https://doi.org/10.32815/jitika.v19i2.1185

Volume 19, Issue 2, 2025

Jurnal Ilmiah Teknologi Informasi Asia

Journal Homepage: https://jurnal.asia.ac.id/index.php/jitika

Article

# Development of an online RIASEC test web application at bakat.uam.ac.id using the Agile Scrum method

Warna Agung Cahyono

*Rekayasa Perangkat Lunak, Universitas Anwar Medika, Sidoarjo, 61263, Indonesia*

***Abstract*— The modern digital era has created a complex job market, increasing the need for accurate career assessment tools. The RIASEC (Realistic, Investigative, Artistic, Social, Enterprising, Conventional) test is a widely recognized framework for aligning personal characteristics with suitable work environments. However, conventional software development methods often struggle to accommodate the continuous refinement required by digital transformation and evolving user needs. This study addresses these challenges by developing an online RIASEC test web application using the Agile Scrum methodology. The research aimed to develop a flexible Scrum framework tailored to this specific application and assess its impact on development speed. The process involved three sprints, with features defined as user stories and prioritized using MoSCoW (Must-have, Should-have, Could-have, Won't-have) rules. The results showed that the durations of sprints 1, 2, and 3 were 2 weeks, 3 weeks, and 3 weeks, respectively, with a total completion time of approximately 269 hours, 7 hours ahead of the estimated 276 hours. The study concludes that Agile Scrum is an effective approach for developing scientific web-based instruments, as it provides a structured yet flexible process that accommodates iterative feedback and successfully mitigates project risks.**

***Keywords*—moscow rules; riasec; scrum; user story; web application;**

## 1. Introduction

The modern digital era is marked by a shifting employment landscape driven by various factors, such as technological innovation, geopolitical dynamics, government policies, and social changes. This phenomenon causes career disruption, creating an increasingly complex job market. In this context, interest and talent assessment instruments serve as a crucial foundation for individuals to make appropriate career decisions. The RIASEC (Realistic, Investigative, Artistic, Social, Enterprising, Conventional) test, developed based on John L. Holland's theory, has been widely recognized as an effective framework for mapping personal characteristics to suitable work environments (Armstrong et al., 2003; Pramudawardani & Adiati, 2024).

However, evolving user needs and digital transformation demand the continuous improvement of this instrument, which has not been fully accommodated by conventional development methods. Web platforms offer strategic advantages over conventional methods because they can reduce the calculation time for psychologists and decrease errors from manual testing (Farid et al., 2021). This makes such platforms attractive for developing an RIASEC test application, which includes interactive capabilities, test history, adaptation of psychometric test items, and real-time reporting. Nevertheless, its development requires multidisciplinary synergy among psychometricians, system designers, and backend developers.

There are various approaches to web development

* Corresponding author.
E-mail Address: warna.agung@uam.ac.id (W.A. Cahyono)

Author E-mail(s): WAC (warna.agung@uam.ac.id)
Digital Object Identifier 10.32815/jitika.v19i2.1185
Manuscript submitted 24 July 2025; revised 6 August 2025; accepted 6 August 2025.
ISSN: 2580-8397(O), 0852-730X(P).

process, which can generally be classified into two main categories: linear, such as the Waterfall model, and adaptive, such as Agile methodology. The Waterfall approach tends to be rigid, especially in the context of developing psychological testing websites, where each stage—from design to coding to validation—is executed separately and without room for iteration. Consequently, changing user needs that emerge mid-development often cannot be accommodated until the next cycle (Istiqomah et al., 2024). This not only prolongs the product's time-to-market but also increases the risk that the resulting product will be outdated upon release. Furthermore, a misalignment between psychometric requirements and technical implementation can result in invalid or less user-friendly applications.

Alternatively, Agile approaches like Scrum offer a revolutionary framework, combining flexibility with strict control through development cycles called sprints. Each sprint is realized in a short period, between 2 to 4 weeks (Nugraha et al., 2023), which in the context of this RIASEC web test development allows for several things: (1) testing specific features, (2) partial psychometric validation, and (3) user experience (UX) improvements based on direct feedback. Cross-disciplinary communication within the team is ensured through practices such as daily stand-ups and sprint reviews.

These principles are rooted in the "Manifesto for Agile Software Development," which prioritizes customer satisfaction through the regular and continuous release of valuable software, openness to changing requirements even in late stages, and intensive collaboration between business stakeholders and the development team. This methodology also emphasizes the importance of building a supportive and motivating work environment for developers and prioritizes face-to-face discussions as a more effective communication method compared to relying on documentation (Cohn, 2011).

By adopting Agile Scrum, this study aims to (1) build a flexible Scrum framework for the development of the RIASEC web test and (2) measure its impact on development speed. The expected practical implications are a reduction in development time and an increase in accuracy in meeting user needs through continuous feedback. Theoretically, this research extends the application of Scrum to the domain of web-based scientific instrument development.

The success of this project is expected to facilitate the digital implementation of career tests for educational institutions and demonstrate that an Agile approach can be adopted for psychodiagnostic applications. This flexibility is important, considering the nature of the RIASEC test, which requires adaptation to local characteristics (Tasrif, 2022), so that question and answer parameters must be changeable quickly without reprinting test documents.

## 2. Method

Non-Agile development methods, such as Waterfall, the spiral model, win-win, and the unified process, are often considered time-inefficient (Alshammari, 2022). These approaches are rigidly sequential, requiring stages such as requirements identification, project planning, system design, implementation, and integration to be completed sequentially. In contrast, Agile methodologies offer a more adaptive approach. Development teams don't need to wait for the entire planning process to be completed, but can instead focus on specific tasks that come up gradually and evaluate the product in short cycles (sprints).

As illustrated in Fig. 1, the Agile Scrum method has a shorter iteration cycle, encompassing product requirements, product planning (in the form of a product backlog), the sprint, and review/release. Agile is a software development approach that focuses on delivering value to the customer iteratively and incrementally, and is highly responsive to changing requirements (Suwarno & Jaya, 2022). In essence, Agile emphasizes close collaboration with the customer, regular delivery of high-value features, and the ability to adapt to market changes, unlike traditional approaches that adhere to rigid long-term plans. The goal is to reduce risk, improve product quality, and ensure the developed solution truly addresses user problems. Therefore, this study adopts the Agile Scrum method with an iterative model implemented through three sprints, each lasting 2-3 weeks.

The Scrum framework itself is built on three empirical pillars: Transparency, where all aspects of the process must be visible and understood by all; Inspection, which is the regular examination of progress to detect deviations; and Adaptation, which means immediate adjustments are made if deviations or changes in requirements occur (Rodriguez et al., 2016). The main principles of Scrum include working in time-boxed iterations (Sprints), managing requirements through a Product Backlog prioritized by Product Owner; a small, cross-functional, and self-organizing team; and the role of a Scrum Master who facilitates the process and removes obstacles for the team.

Broadly, the Scrum stages in this study are illustrated in Fig. 2 and are divided into three main parts: the pre-sprint stage, the sprint stage, and the evaluation stage, in the form of a sprint retrospective (Rad, 2021). The recurring development cycle occurs in the sprint phase.

The first stage, or pre-sprint, begins with the formulation of a vision statement from the user's perspective and the creation of a product roadmap by the Product Owner as a timeline for feature development. After that, a Scrum Team is formed, consisting of multidisciplinary roles: Scrum Master, Product Owner, and Developer. The team then compiles the Product Backlog, which is a list of product requirements. These requirements are gathered from users and formulated into a user story format by the Product Owner (Cohn, 2011). The backlog for this application includes key features such as authentication, a question bank, the RIASEC scoring model, a results dashboard, and an administration module. One of the advantages of Scrum is its flexibility, which does not require all user stories to be completed before moving to the next stage.

Before entering the sprint phase, it is necessary to prioritize the items in the Product Backlog. This study uses the MoSCoW (Must-have, Should-have, Could-have, Won't-have) method to prioritize items for inclusion in the sprint based on risk analysis and business value (Hermans, 2023; Miranda, 2022). In this context, features in the Must-have category focus on content validity and data security, for both admins and participants.

The next stage is the sprint execution, which begins with sprint planning to break down each selected product backlog item into smaller tasks. The implementation process involves Git-based development and daily synchronization through a 15-minute daily stand-up to discuss progress and obstacles. At the end of each sprint, the resulting product increment is validated through two mechanisms (Agustini, 2023). First, a Sprint Review involving stakeholders (such as institutional public relations

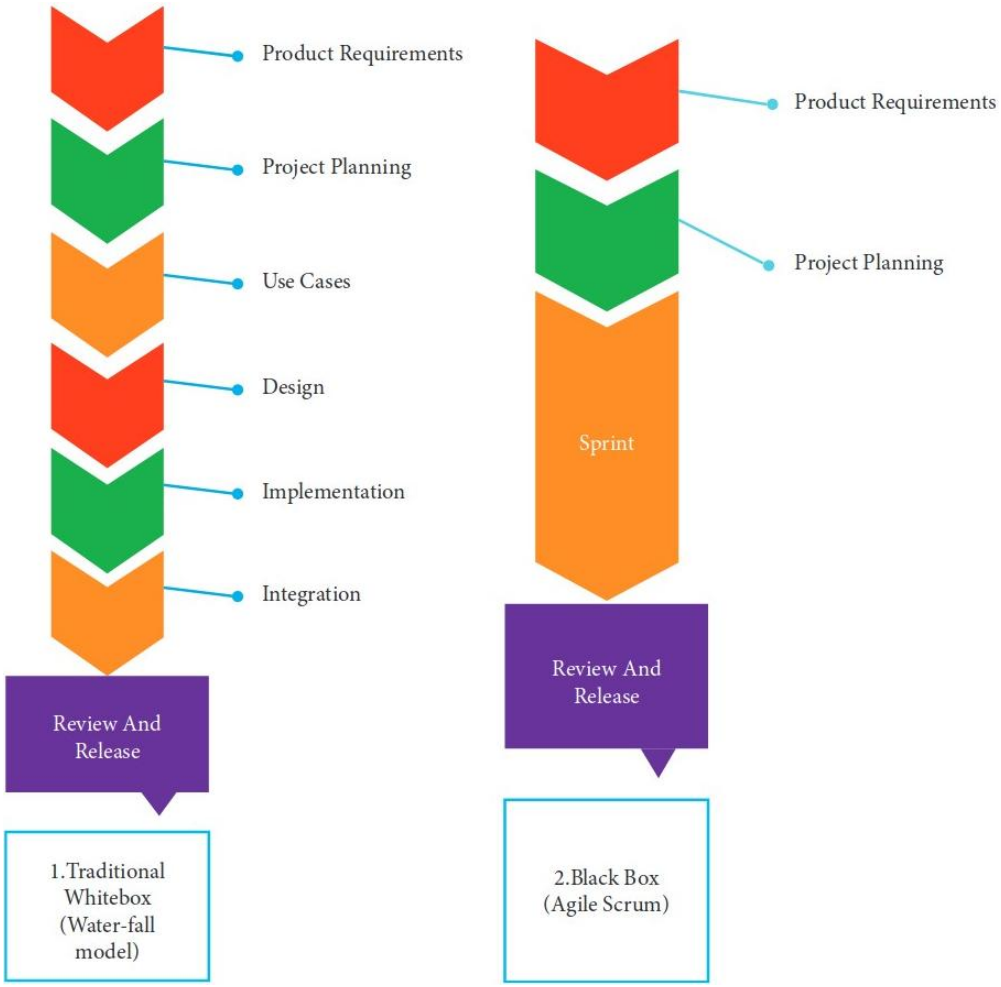



Fig. 1. Analogy comparison of Waterfall and Agile Scrum

and potential users) to test functionality and user experience. Second, a Psychometric Sprint Review is conducted by a psychometrician or doctor to verify the validity of the test items and the consistency of the test results. Finally, the feedback obtained from both review sessions becomes the main input for the Sprint Retrospective. In this forum, the team conducts an evaluation to refine the work process for the next sprint (Setiawan et al., 2024).

Based on Fig. 3, the application of MoSCoW rules allows the team to manage time-related risks. The Should-have category can function as a buffer to anticipate delays in the completion of Must-have items that do not align with initial predictions. Furthermore, as illustrated in Fig. 2, the sprint phase includes a series of structured activities. This process begins with sprint planning, a meeting where the team plans the work to be done in one sprint cycle (e.g., a fixed duration of 2 weeks) based on priorities calculated using MoSCoW rules. In this meeting, the Product Owner determines the essential requirements from the user stories to be included in the Sprint Backlog. Then, these stories are broken down into more technical and detailed tasks.

At the end of the sprint, the team conducts a sprint demo to demonstrate the functionality that has been successfully developed according to the targeted user stories. This stage also includes a sprint review, which forms the basis for discussion in the sprint retrospective. In the retrospective forum, the team evaluates the review results to decide if improvements are needed in the work process for the next sprint. The entire software development process is guided and directed by the Scrum Master, who ensures that all involved parties perform their roles effectively.

## 3. Results

This section presents the results and discussion from the application of the Scrum methodology in the development of the RIASEC test application, which is accessible at https://bakat.uam.ac.id. Following the flow in Fig. 2, the Agile Scrum process in this study was structured into three main phases: pre-sprint, sprint execution, and final evaluation. The research systematics included:

a. the pre-sprint stage, which involved formulating the vision statement, creating the product roadmap, developing user stories, and creating the product backlog;
b. the sprint stage, which consisted of planning the sprint backlog, executing sprint tasks, and conducting the sprint demo and review; and
c. concluded with the evaluation stage through a sprint retrospective, with data taken from sprint tasks to create a burndown chart.

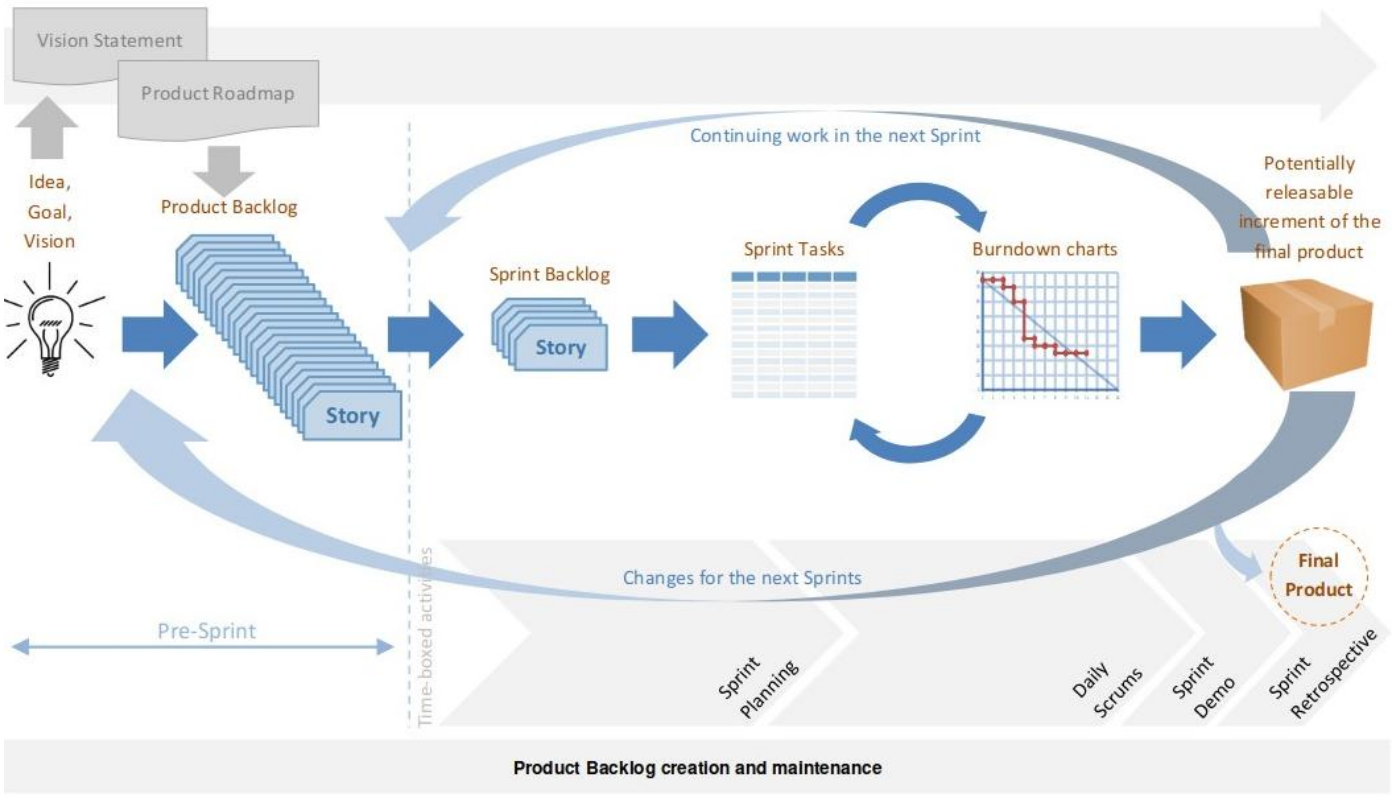



Fig. 2. Stages of scrum implementation

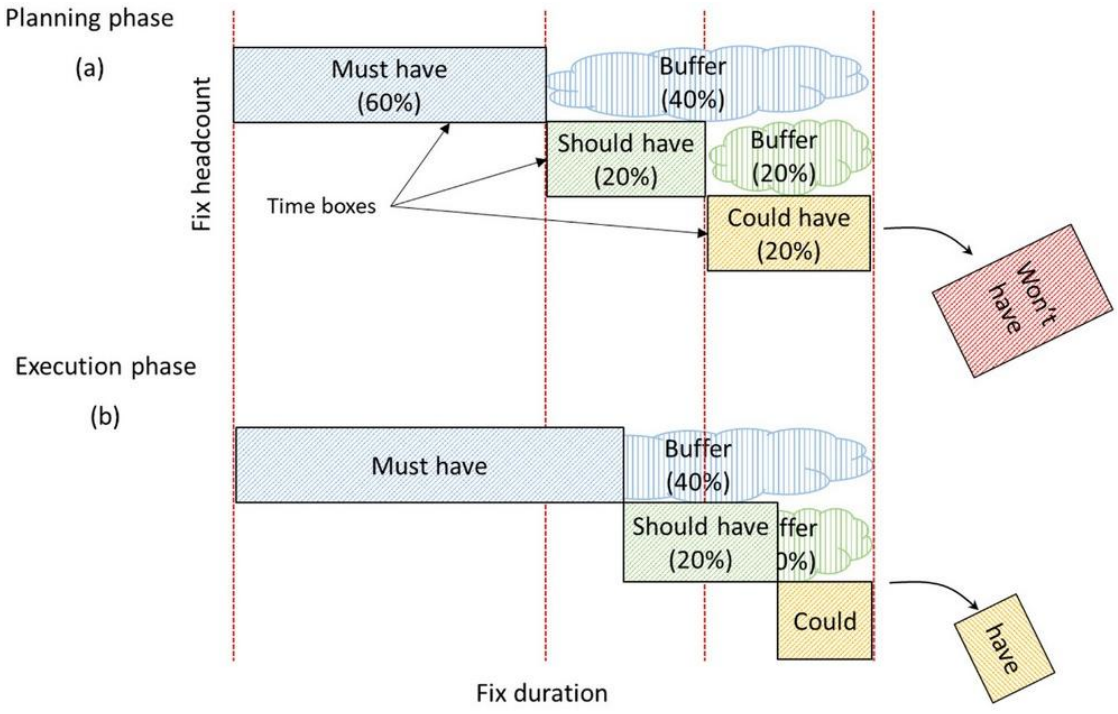



Fig. 3. MoSCoW rules pattern with should-have as a buffer

### 3.1. Pre-sprint

The pre-sprint stage is the initial preparation phase in the Scrum process. This phase begins with the formulation of a vision statement, which is a description of the product definition from the user's perspective. For this project, the product definition and its vision statement are as follows:

- Product: RIASEC aptitude and interest test application aimed at upper-level students
- Vision Statement: Provide an online testing platform that can be accessed anywhere, ensuring that test progress is saved even if the internet connection is lost. This application aims to provide a map of students' personal characteristics and provide an overview of future career potential for Guidance and Counseling (BK) teachers.

Based on this vision statement, a product roadmap was created, with its global strategy detailed in Table 1. It should be noted that the timeline and phases in this roadmap are flexible. Adjustments can occur during the process of refining the Product Backlog into user stories, and the timeline may change again after calculating the ratio of value and effort, as will be explained in Table 4.

Table 1. Product Roadmap

| Stage | Epic | Goal |
|---|---|---|
| First 2-weeks | Student online test | Students can take the test, and if the connection is lost, they can continue without starting over. |
| Second 2-weeks | Test Results | Students can view their test results, while guidance counselors can also view the results of students at their school. |
| Third 2-weeks | Test History | Students and administrators can view the history of tests taken. |

### 3.2. User story

The system design process began with the identification of functional requirements, which in this study resulted in 14 structured requirements. These requirements were formulated into the user story format after an observation and interview process with five stakeholder groups: a psychometrician (doctor), Universitas Anwar Medika (UAM) public relations, the foundation's IT team, the head of LPTIK (*Lembaga Pengembangan Teknologi Informasi dan Komunikasi*), developers, and students.

Table 2. List of user story

| Code | User story description |
|---|---|
| User with the student role: | |
| US1 | As a student, I want to create an account so I can log in. |
| US2 | As a student, I want to log in so I can enter the system. |
| US3 | As a user, I want to start the test to understand my personal characteristics. |
| US4 | As a user, I want to see the test results so my strengths and weaknesses can be mapped. |
| US5 | As a user, I want to see my test history to check for differences in characteristics at different times. |
| US6 | As a user, I want to log out so my phone/laptop can be used by another user. |
| US6b | As a user, I can click on a major description to get information about that major. |
| User with the admin role: | |
| US7 | As an admin, I want to log in so I can get in. |
| US8 | As an admin, I want to see the list of students to know if they have created an account. |
| US9 | As an admin, I want to see the test history per student to know their potential. |
| US10 | As an admin, I want to be able to add new school data that is not yet registered in the online RIASEC test system. |
| US11 | As an admin, I want to create a share link for test history for school public relations so the school can see student test reports for a certain period. |
| US12 | As an admin, I want to log out so that after use, no one else can enter without logging in. |
| US12b | As an admin, I want to broadcast RIASEC test results to specific users. |

### 3.3. Product backlog

The collected user stories were then used to refine and form the Product Backlog. The user story format ("As a [role], I want [goal], so that [outcome]") proved to be very effective for gathering user requirements in detail, although this study does not further discuss user story cards. The collection of user stories in Table 2 was then processed into more concrete Product Backlog Items (PBIs), as detailed in Table 3.

### 3.4. Sprint backlog

After the list of PBIs was compiled, the next stage was sprint planning to form the Sprint Backlog. In this phase, a prioritization process was conducted to determine which PBIs would be included in the next sprint. This prioritization was based on a score ratio calculated between the value from a business and user perspective and the estimated effort from the development team's side, as shown in Table 4.

After each PBI was given an initial score (as in Table 4), the next step was to calculate the ratio between Value and Effort to prioritize the PBIs objectively. In this context, Value is defined as the value of a feature from the user's perspective, while Effort is the estimated work required by the development team to realize it (Del Sagrado & Del Águila, 2021). The basic principle is that a feature's priority is directly proportional to its value and inversely proportional to its development effort, as shown in Table 5. This relationship can be formulated as follows:

$$\text{ratio}_{\text{value}_{\text{effort}}} = \frac{\text{value}}{\text{effort}}. \quad (1)$$

Using this formula, the ratio for each PBI was calculated, and the results are presented in Table 6. Based on this ratio score, each Product Backlog Item was then classified using MoSCoW rules (Must-have, Should-have, Could-have, Won't-have), as presented in Table 7.

Must-have are features that must be present for the system to function at least minimally. Without these features, the product is considered inoperable. An example in this project is

Table 3. PBIs for the RIASEC Test at https://bakat.uam.ac.id

| Code | PBI Description | Notes |
|---|---|---|
| PBI.1 | User Login/Register/Logout | Combination of US1, US2, and US6. |
| PBI.2 | Test questions | US3 is a feature for a multiple-choice (yes/no) test. |
| PBI.3 | Test Result Graph | US4 is a circular radar chart showing the dominance of self-characteristics. |
| PBI.4 | Test history | US5 is a list of the history of similar tests taken on this application (bakat.uam.ac.id). |
| PBI.5 | Admin login/logout | Combination of US7 and US12. |
| PBI.6 | List of students and student test history | US9 allows the admin of bakat.uam.ac.id (i.e., UAM public relations) to check if a student has created an account, and to see their test status and potential. |
| PBI.7 | Add school | US10 allows the admin to add school data if a student cannot register because their school is not listed. |
| PBI.8 | Create test history link for a specific period per school | US11 allows the admin to generate a special link for schools needing test results from a certain period. |
| PBI.9 | Description of majors based on RIASEC results | After the user takes the test, the results page has a link to descriptions of suitable majors. |
| PBI.10 | Broadcast results via WhatsApp | The admin wants to send results for some, or all tests taken by users. |

Table 4. Scores for business value and development effort (1-5)

| PBI | Value (V) | Effort (E) | Estimate(hours) |
|---|---|---|---|
| PBI.1 | 5 | 1 | 12 |
| PBI.2 | 5 | 3 | 36 |
| PBI.3 | 5 | 4 | 48 |
| PBI.4 | 3 | 3 | 36 |
| PBI.5 | 4 | 1 | 12 |
| PBI.6 | 3 | 4 | 48 |
| PBI.7 | 5 | 2 | 12 |
| PBI.8 | 5 | 4 | 48 |
| PBI.9 | 1 | 2 | 24 |
| PBI.10 | 1 | 5 | 60 |
| Total | | | 336 |

Table 5. MoSCoW pattern

| | Category | Description |
|---|---|---|
| M | Must Have | High Value vs. Low Effort |
| S | Should have | High Value vs. High Effort |
| C | Could have | Low Value vs. Low Effort |
| W | Won't have | Low Value vs. High Effort |

Table 6. Sorted ratio of value and effort

| PBI | value (v) | effort (e) | Ratio | Estimate (hours) |
|---|---|---|---|---|
| PBI.1 | 5 | 1 | 5 | 12 |
| PBI.5 | 4 | 1 | 4 | 12 |
| PBI.7 | 5 | 2 | 2,5 | 12 |
| PBI.2 | 5 | 3 | 1,67 | 36 |
| PBI.3 | 5 | 4 | 1,25 | 48 |
| PBI.8 | 5 | 4 | 1,25 | 48 |
| PBI.4 | 3 | 3 | 1 | 36 |
| PBI.6 | 3 | 4 | 0,75 | 48 |
| PBI.9 | 1 | 2 | 0,5 | 24 |
| PBI.10 | 1 | 5 | 0,2 | 60 |

Table 7. MoSCoW rules for product classification

| Classification | List of PBIs |
|---|---|
| M | PBI.1, PBI.2, PBI.5, PBI.7 |
| S | PBI.3 dan PBI.8 |
| C | PBI.4, PBI.6, PBI.9 |
| W | PBI.10 |

Table 8. Total estimate after eliminating won't have

| | List of PBIs | Estimate | Percentage |
|---|---|---|---|
| M | PBI.1, PBI.2, PBI.5, PBI.7 | 72 hours | 26 % |
| S | PBI.3 dan PBI.8 | 96 hours | 35 % |
| C | PBI.4, PBI.6, PBI.9 | 108 hours | 39 % |
| Total | | 276 hours | 100 % |

the core functionality of the RIASEC test questions.

Should-have features are essential and should be present, but they aren't critical. The application can still run without them, although their functionality will be less complete. An example is a radar chart of test results that maps user characteristics.

Could-have features are features that are expected to be present because they add value, but are not essential. An example is the feature to view test history.

Won't-have represents a feature that, at this stage, was decided not to be developed, although it might be needed in the future. An example is the feature to broadcast test results via WhatsApp. This feature was not prioritized because, based on feedback, guidance counselors preferred students to receive test results directly from them, not through automatic system notifications.

This classification became the basis for formulating which PBIs would be included in the Sprint Backlog during the sprint planning stage, as shown in Table 8.

Table 9. Sprint 1: duration 2 weeks, estimate 72 hours

| PBI | Item | Est. | SBI | Item | Run |
|---|---|---|---|---|---|
| PBI.1 | Participant authentication (register, login, logout) | 12 | 01 | Implemented authentication using Laravel/Breeze scaffolding. | 1 |
| | | | 02 | Adjusted the user registration page layout. | 3 |
| | | | 03 | Created landing and login pages for participants. | 4 |
| | | | 04 | Implemented participant logout functionality. | 1 |
| Estimation | | 12 | Total | | 9 |
| PBI.5 | Admin authentication (login & logout with role guard). Guard role is used to fetch data from a different user database table. | 12 | 05 | Configured guard (auth.php) and middleware (admin.php) for admin role authentication. | 5 |
| | | | 06 | Created the admin dashboard page. | 5 |
| | | | 07 | Implemented logout functionality specific to the admin role. | 2 |
| Estimation | | 12 | Total | | 12 |
| PBI.7 | School data management | 12 | 08 | Created a data table to manage school data across Indonesia. | 7 |
| | | | 09 | Implemented a form to add new school data. | 5 |
| Estimation | | 12 | Total | | 12 |
| PBI.2 | Multiple-choice test with RIASEC questions | 36 | 10 | Created the test instructions page. | 2 |
| | | | 11 | Implemented session handling to maintain test progress during connection loss. | 4 |
| | | | 12 | Created data models for RIASEC questions and categories. | 5 |
| | | | 13 | Developed an API for randomizing test questions. | 6 |
| | | | 14 | Designed the user interface layout for displaying questions. | 5 |
| | | | 15 | Developed an API to save answers and retrieve the next question (SBI 13). | 5 |
| | | | 16 | Implemented functionality to save final test results and end the session. | 6 |
| Estimation | | 36 | Total | | 33 |
| Total Sprint 1 | | 72 | Total Sprint 1 | | 66 |

Table 10. Sprint 2: duration 3 weeks, estimate 96 hours

| PBI | Item | Est. | SBI | Item | Run |
|---|---|---|---|---|---|
| PBI.3 | Visualization of test result graph | 48 | 17 | Calculated test scores per category into statistical data (JSON). | 20 |
| | | | 18 | Created label cards for category descriptions and major recommendations. | 15 |
| | | | 19 | Created a radar chart for visualizing the classification of test results. | 15 |
| Estimation | | 48 | Total | | 50 |
| PBI.8 | Report sharing feature for guidance counselors | 48 | 20 | Implemented report filters by school and time period. | 10 |
| | | | 21 | Implemented functionality to select data (SBI 20) and generate a report link. | 17 |
| | | | 22 | Created a list page of links displaying student test results (including 3 main characteristics and suitable majors). | 10 |
| | | | 23 | Created a detailed report page showing explanations of the 6 characteristics and a radar chart. | 10 |
| Estimation | | 48 | Total | | 47 |
| Total Sprint 2 | | 96 | Total Sprint 2 | | 97 |

The sprint planning strategy is carried out according to the MoSCoW rule (Fig. 3), where Must-have items are prioritized for work. This approach also utilizes other categories as buffer zones to mitigate the risk of delays. Practically, Should-have items serve as buffers for Must-haves, and Could-have items serve as buffers for Should-haves. This structure is designed so that if delays occur, the impact will first affect lower-priority features.

After the MoSCoW classification was applied to the ten PBIs, the total estimated work time was recalculated by eliminating items from the Won't Have category, as detailed in Table 9. Based on the table, the work allocation was distributed as follows: 26% for Must-have PBIs (PBI.1, PBI.2, PBI.5, and PBI.7) and 35% for Should-have PBIs, which served as the main buffer zone (PBI.3 and PBI.8). The Could-have category became the final buffer. Thus, if a delivery delay occurred, the features most likely to be affected would be the non-critical Could-have items, leaving room for negotiation without sacrificing the core

Table 11. Sprint 3: duration 3 weeks, estimate 108 hours

| PBI | Item | Est. | SBI | Item | Run |
|---|---|---|---|---|---|
| PBI.4 | Participant-side test history | 36 | 34 | Created a detailed test history page (including 3 main characteristics and majors). | 18 |
| | | | 25 | Implemented a radar chart on the test history page. | 17 |
| Estimation | | 36 | Total | | 35 |
| PBI.6 | Admin-side list of students and student history | 48 | 26 | Created a data table for the student list with pagination. | 14 |
| | | | 27 | Implemented functionality to edit, delete, hide, and disable student accounts. | 9 |
| | | | 28 | Created a test history list per student on the admin dashboard. | 8 |
| | | | 29 | Implemented label cards for descriptions and major recommendations on the admin page. | 8 |
| | | | 30 | Implemented a radar chart for test result classification on the admin page. | 10 |
| Estimation | | 48 | Total | | 49 |
| PBI.9 | Description of majors/study programs from test results | 24 | 8 | Created a relational data model between RIASEC categories and majors. | 6 |
| | | | 11 | Designed the user interface (UI) to display major descriptions. | 10 |
| | | | 5 | Implemented a hyperlink badge from test results to the major description page. | 6 |
| Estimation | | 24 | Total | | 22 |
| Total Sprint | | 108 | Total Sprint | | 106 |

functionality of the product.

### *3.5. Sprint task*

Once PBI priorities are established using MoSCoW, the development process enters the execution phase, which is the core of the Scrum framework. This phase centers on the execution of sprints, which are flexible, fixed-duration work cycles, ranging from 2 to 4 weeks, tailored to project needs.

To determine the work capacity of each sprint, calculations are made based on the team's effective working hours. Assuming Anwar Medika University has a 40-hour work week with a 1-hour break per day; the total effective work week is 35 hours (or 40 work hours minus 5 hours of rest per week). Therefore, a two-week sprint has a capacity of approximately 70 hours. Within this duration, the prioritized PBIs (by eliminating the "Won't Have" category for efficiency in the Table 10) are broken down into smaller, more manageable task units, referred to in this study as Sprint Backlog Items (SBIs).

Any deviation between the actual sprint task completion and the estimate set in the Sprint Backlog will be the primary evaluation material in the Sprint Retrospective session. This analysis is very useful for improving the planning process in the next sprint cycle and future projects. Based on the final PBI priorities, SBIs are compiled as shown in Table 9 to Table 11. Furthermore, burndown charts will be displayed in the discussion chapter (Fig. 13).

### *3.6. Sprint demo*

The Sprint Review phase is an evaluation session held at the end of each sprint to demonstrate the work completed by the development team (Pratama et al., 2022). The primary goal of this phase is to identify defects or functional inconsistencies, with each finding contributing to the solution found during the Sprint Retrospective session (Lee & Chen, 2023).

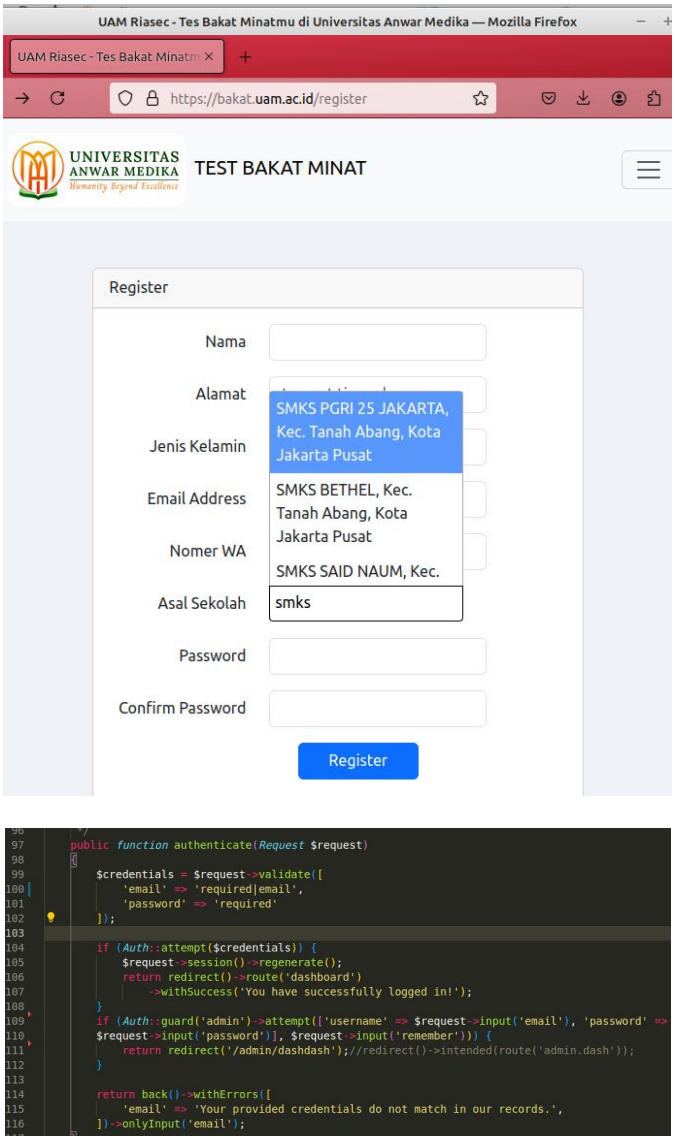
Table 12. Review to find defects

| PBI | Description | Review Result |
|---|---|---|
| PBI.1 | User Login/Register/Logout | Minor revision |
| PBI.2 | Test questions | Accepted |
| PBI.3 | Test Result Graph | Minor defect |
| PBI.4 | Test history | Accepted |
| PBI.5 | Admin login/logout | Accepted |
| PBI.6 | List of students and student test history | Accepted |
| PBI.7 | Add school | Accepted |
| PBI.8 | Create test history link for a specific period per school | Accepted |
| PBI.9 | Description of majors/study programs based on RIASEC results | Accepted |

The results of the review process conducted on the three sprints in this study are summarized in Table 12. One significant finding from the first sprint was related to the user registration process. It was found that the number of mandatory fields was too high, potentially reducing users' interest in registering. As a solution, the development team simplified the registration form by only requiring essential data: email, name, school of origin, and WhatsApp number. Fig. 4 through Fig. 12 display functional reviews of the nine implemented PBIs, along with some relevant technical details.

In the registration feature (Fig. 4), the list of schools a user can select is limited to those they have already registered with. Meanwhile, the user login process utilizes the default authentication of the Laravel framework.

For the core functionality, namely test execution, a mechanism is implemented to maintain data integrity and user convenience. As shown in Fig. 5, a code snippet in line 98

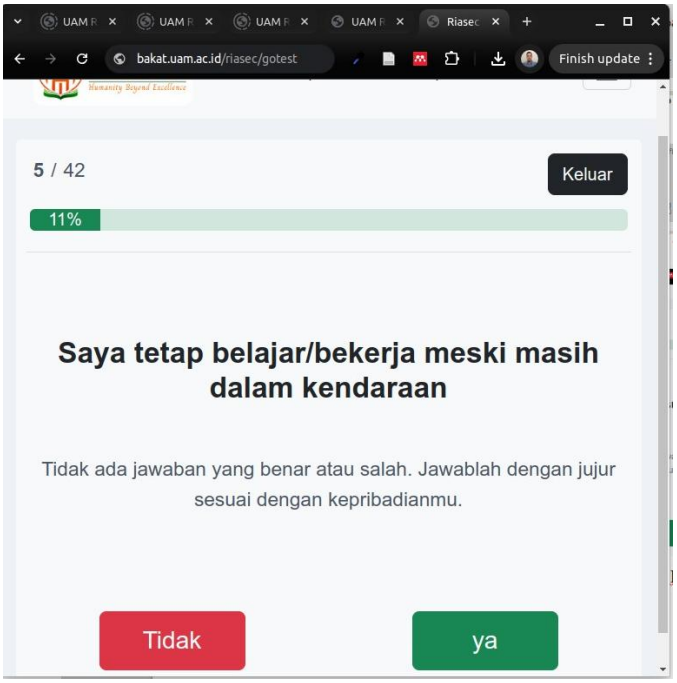



```
public function authenticate(Request $request)
{
    $credentials = $request->validate([
        'email' => 'required|email',
        'password' => 'required'
    ]);

    if (Auth::attempt($credentials)) {
        $request->session()->regenerate();
        return redirect()->route('dashboard')
            ->withSuccess('You have successfully logged in!');
    }
    if (Auth::guard('admin')->attempt(['username' => $request->input('email'), 'password' =>
    $request->input('password')], $request->input('remember'))) {
        return redirect('/admin/dashdash');//redirect()->intended(route('admin.dash'));
    }

    return back()->withErrors([
        'email' => 'Your provided credentials do not match in our records.',
    ])->onlyInput('email');
```

Fig. 4. PBI.1 register and login




```
private function getriasec($id)
{
    $ases = array_keys(Session::get('riasec')[1]);
    $aSoal = (new SoalRia)::whereNotIn("id_soal", $ases)->pluck("id_soal");
    $idx = rand(0, count($aSoal) - 1);
    $oSoal = (new SoalRia)::where("id_soal", count($aSoal) == 0 ?
                        $ases[count($ases) - 1] : $aSoal[$idx])->first();
    return response(json_encode(["code" => 0, "alert" => "Berhasil",
            "message" => "Get Riasec", "showalert" => false,
            "data" => json_encode(array("soal" => $oSoal,
            "progress" => count($aSoal) == 0 ? count($ases) : count($ases) + 1,
            "total" => count($ases) + count($aSoal))),
            201)->header('Content-Type', 'application/json');
}
```

Fig. 5. PBI.2 RIASEC test

ensures that test progress will not be lost if the internet connection is lost during an active test session. Furthermore, the *whereNotIn* method implemented in line 99 ensures that the same test question will not be presented to the user repeatedly.

The test result display, as illustrated in Fig. 6, was refined based on feedback from the review stage. This page presents a description of the student's psychological characteristics at the top and a data visualization in the form of a radar chart at the bottom. The test history feature was also successfully implemented and functions correctly, as seen in Fig. 7. This page displays a list of tests that have been taken and includes a button to view detailed results, which is directly linked to the graph visualization in PBI.3.

To accommodate administrative needs, a series of special features was developed on the admin side. The admin authentication process is separated from that of regular users by adding a special guard to the Laravel framework to handle the different user table locations, as illustrated in the Fig. 8.

From the admin dashboard, user and data management functionalities can be accessed. The admin can view the list of students along with their test history through an interactive interface (Fig. 9). In addition, there is also a feature for managing school data, which will be displayed on the registration page (Fig. 10).

One of the key features is the ability to generate a report link that can be shared with guidance counselors. As shown in Fig. 11, the admin can create a unique URL that gives the school access to view a summary of their students' interests and talents. To supplement the test results, the application also provides a description page for each recommended major. This page can be accessed via a hyperlink available below the test result radar chart (Fig. 12), providing in-depth information for users to further explore their career options.

## 4. Discussion

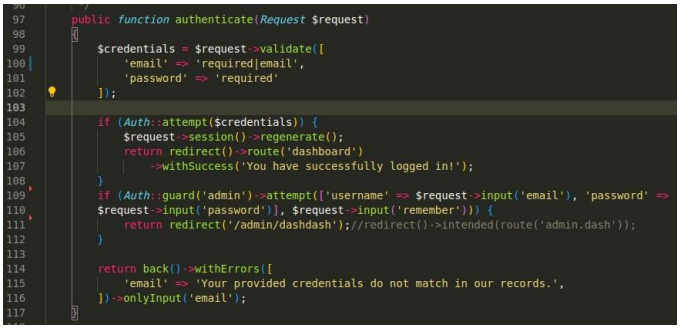
The Sprint Retrospective stage aims to evaluate the entire Scrum process that has been carried out. This session focuses on analyzing the review results from each sprint with the goal of improving team performance in the future. One of the main evaluation tools used is the sprint burndown chart, which compares the estimated work time with its realization.

Overall, the data shows that the total estimated work time for the completed PBIs was 276 hours. However, the actual work from sprints 1 to 3 was recorded at only 269 hours, or 7 hours faster than the initial estimate, as seen in Table 9 through Table 11. Nevertheless, a deeper analysis of the burndown chart (Fig. 13) reveals different speed dynamics in each sprint.



Fig. 6. PBI.3 test result graph

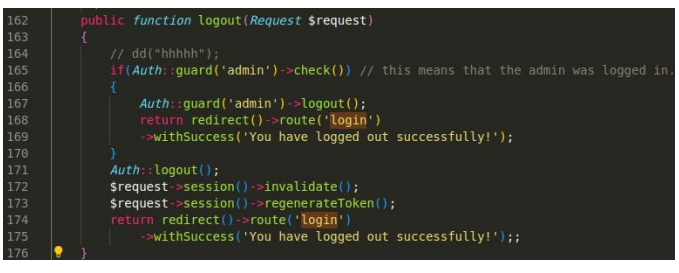

Fig. 7. PBI.4 test history



```
public function authenticate(Request $request)
{
    $credentials = $request->validate([
        'email' => 'required|email',
        'password' => 'required'
    ]);

    if (Auth::attempt($credentials)) {
        $request->session()->regenerate();
        return redirect()->route('dashboard')
            ->withSuccess('You have successfully logged in!');
    }
    if (Auth::guard('admin')->attempt(['username' => $request->input('email'), 'password' =>
    $request->input('password')], $request->input('remember'))) {
        return redirect('/admin/dashdash');//redirect()->intended(route('admin.dash'));
    }

    return back()->withErrors([
        'email' => 'Your provided credentials do not match in our records.',
    ])->onlyInput('email');
}
```



```
public function logout(Request $request)
{
    // dd("hhhhh");
    if(Auth::guard('admin')->check()) // this means that the admin was logged in.
    {
        Auth::guard('admin')->logout();
        return redirect()->route('login')
        ->withSuccess('You have logged out successfully!');
    }
    Auth::logout();
    $request->session()->invalidate();
    $request->session()->regenerateToken();
    return redirect()->route('login')
        ->withSuccess('You have logged out successfully!');;
}
```

Fig. 8. PBI.5 admin login and logout

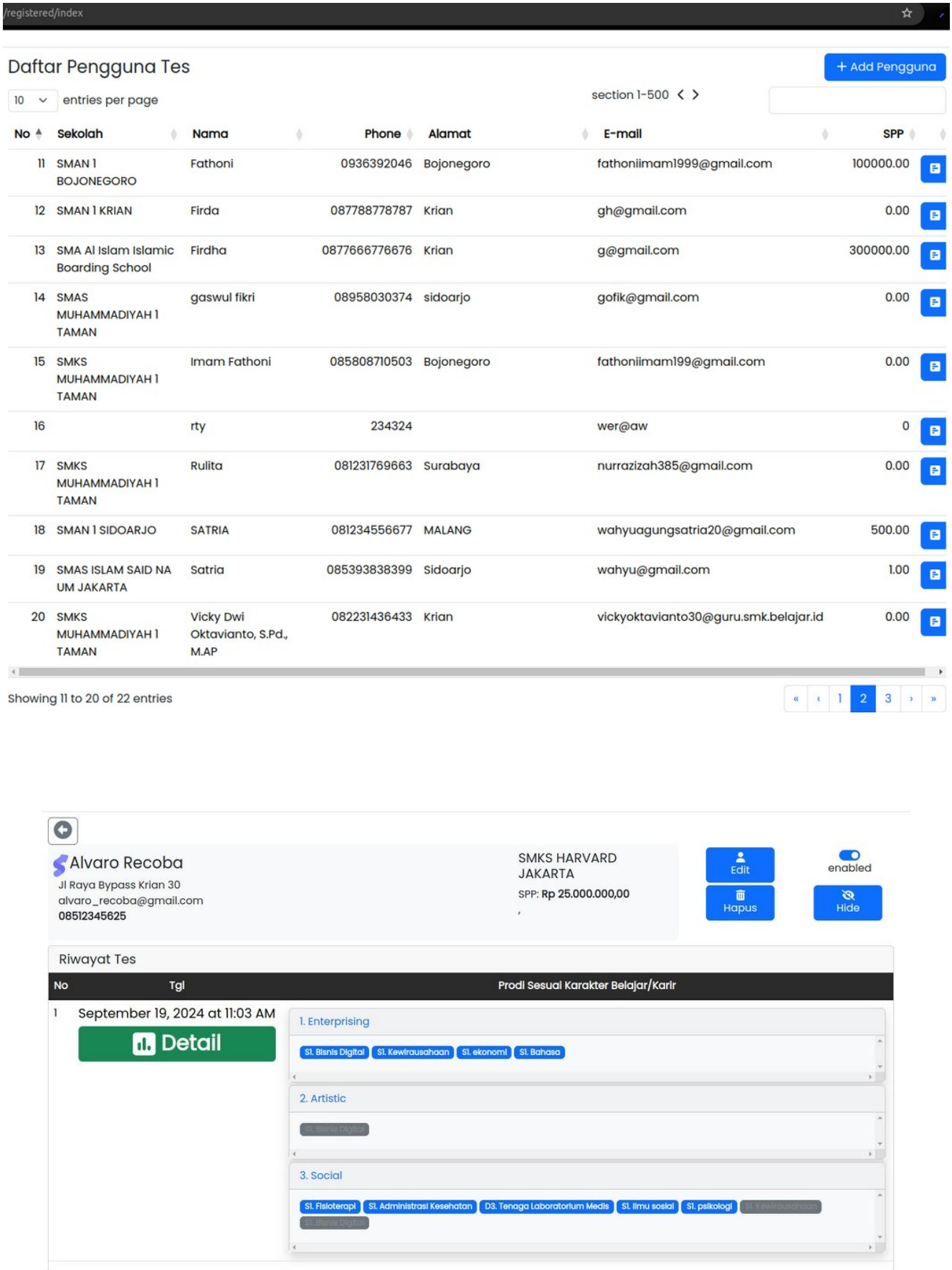



Fig. 9. PBI.6 list of students with test history

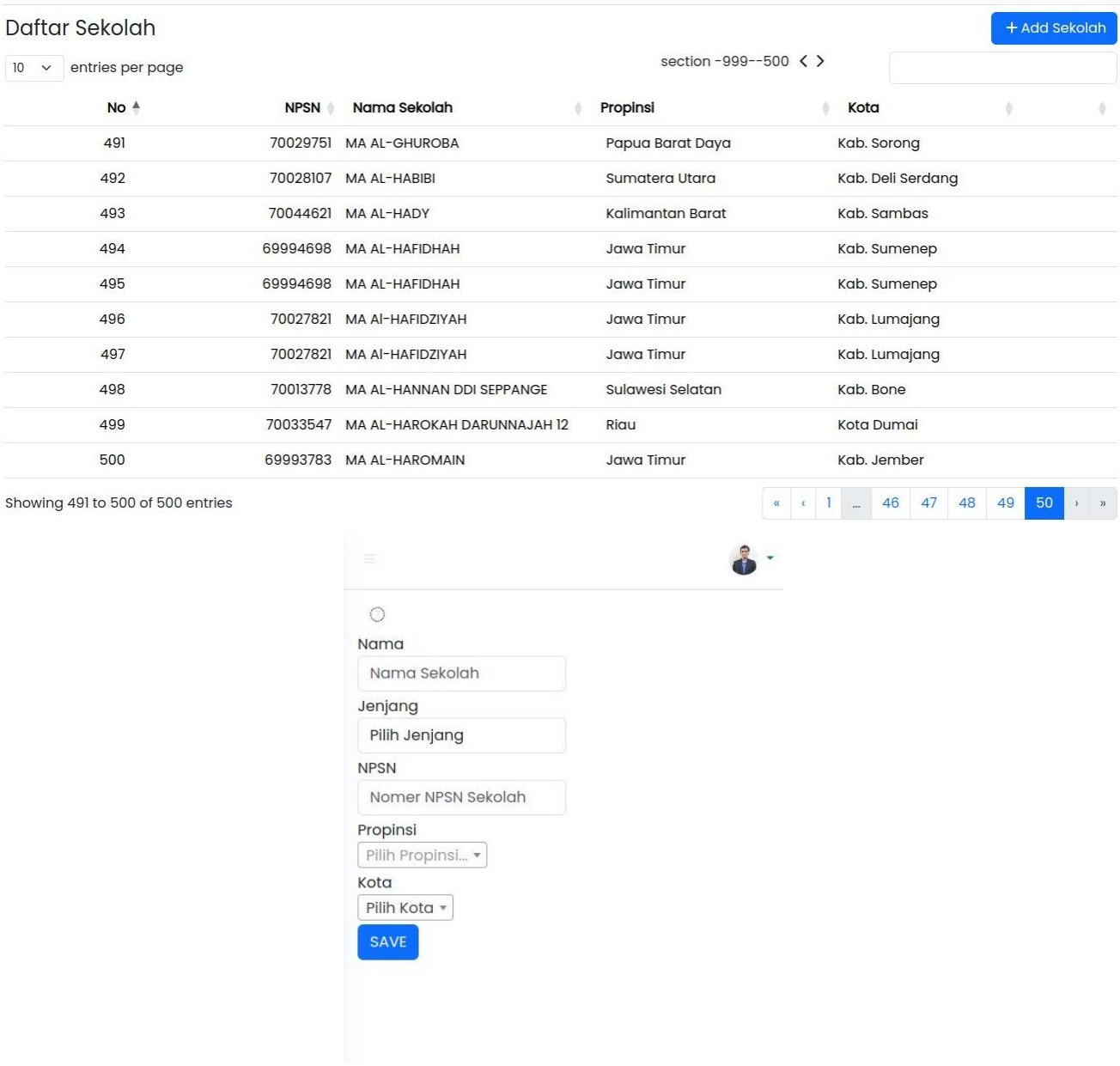



Fig. 10. PBI.7 list of schools and form for managing the school list

In the first sprint, the development team worked 6 hours faster than planned. This difference was mainly due to an inaccurate time estimate for PBI.1 (login, logout, and registration features). This finding is a valuable lesson for similar future projects, where the time estimates for PBI.1 can be reduced to 9 hours and PBI.2 to 33 hours.

Conversely, in the second sprint, the burndown chart (Fig. 13) indicated that the development team worked slower than planned. This may have been caused by inaccuracy in estimating the effort score in the previous stage. Specifically, the work on PBI.3 took 2 hours longer than estimated, although PBI.8 was completed 1 hour faster. Consequently, there was a net delay of 1 hour in this sprint. This finding serves as an evaluation point for future projects of similar difficulty. Furthermore, analysis during the daily scrum sessions identified that the main delay occurred in SBI.17 (test score calculation), which took 15 hours to complete. The cause was the technical difficulty of the score calculation algorithm and a misinterpretation of requirements—particularly regarding the business rule about the maximum number of dominant characteristics to display and how the system should respond if there were more than three. This incident highlights that even with the help of user stories, the level of detail in the Product Owner's translation of user needs into PBIs can still be improved.

In the third sprint, the realization graph was again below the estimate, indicating no delays. This efficient performance was partly due to the completion of error fixes from PBI.1 and PBI.3 that were found in the previous sprint's review. By addressing these targets, the team successfully compensated for the minor delay that occurred in the second sprint.

## 5. Kesimpulan

Based on the research results, the Agile Scrum method proved effective in guiding the team during the software development process for the RIASEC test application. This

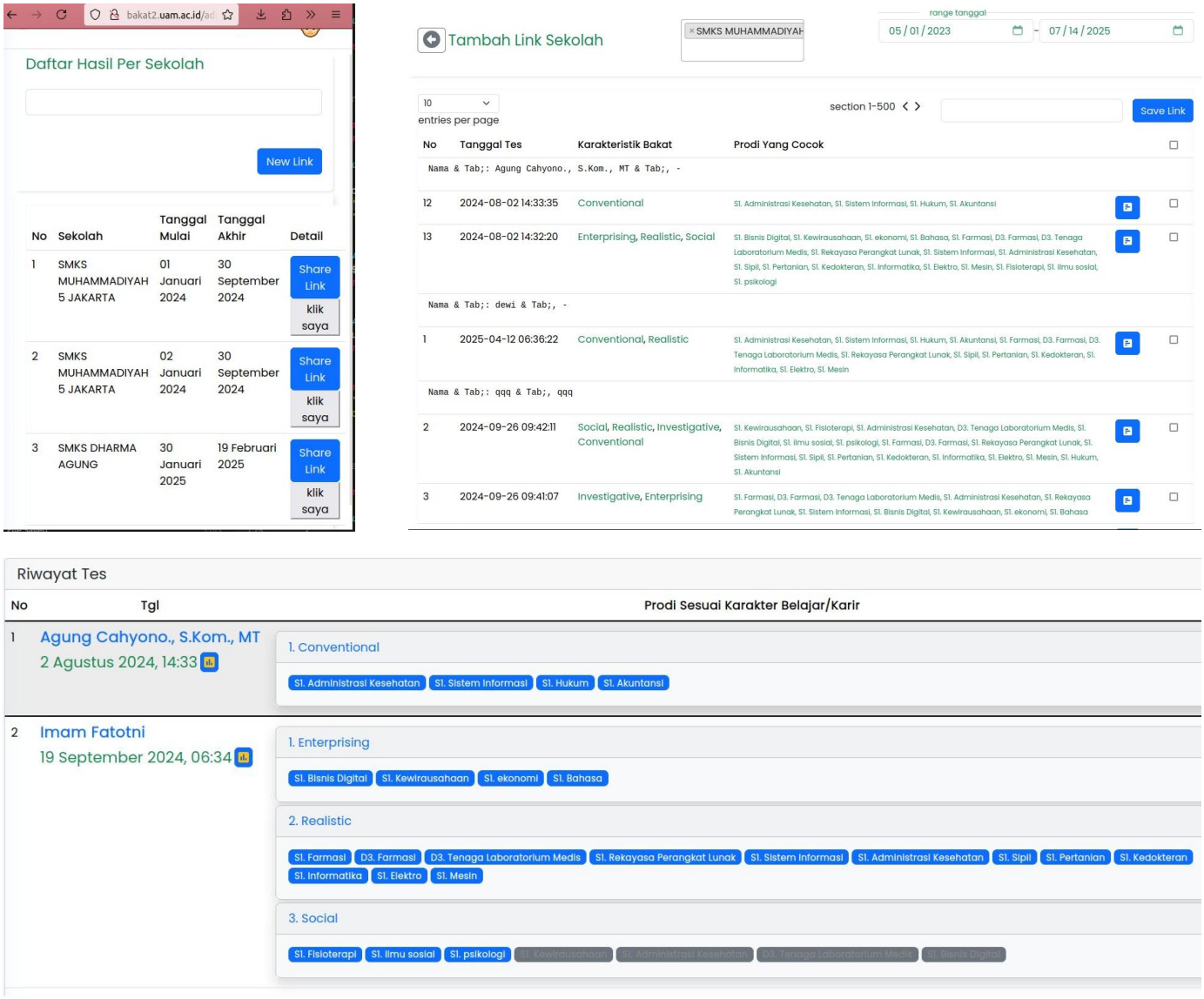



Fig. 11. PBI.8 Share Link for Guidance Counselors (BK teachers)

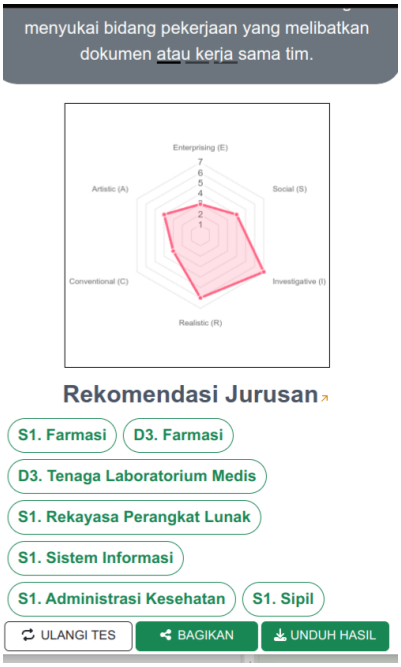




Fig. 12. PBI.9 description of majors/study programs according to RIASEC category

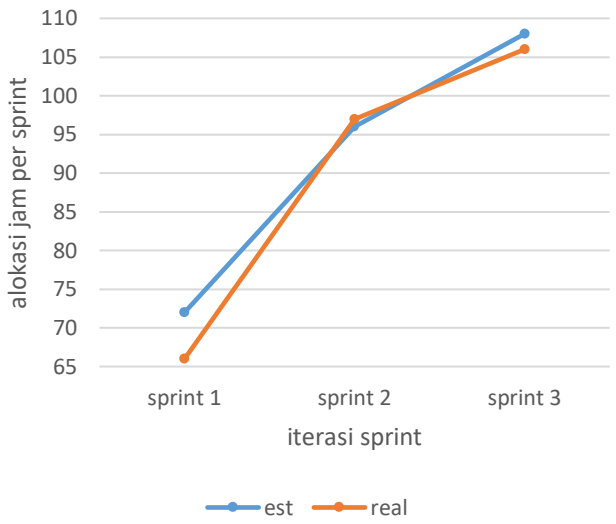



Fig. 13. Overall sprint burndown chart

framework allowed the Product Owner to detail Product Backlog Items (PBIs) effectively using user stories and to prioritize them using the MoSCoW method. Subsequently, during the sprint task stage, the development team could focus on breaking down these PBIs into smaller technical tasks (Sprint Backlog Items or SBIs).

One of the key advantages identified was how Scrum provides autonomy to developers through SBIs, which was shown to encourage creative ideas. Nevertheless, this autonomy remained within a controlled framework, as the entire process was guided by the established PBIs. This created an environment where developers felt free to manage their technical work while remaining aligned with the project's overall goals.

The use of the MoSCoW method also successfully mitigated the risk of delays. Interestingly, the buffer zone theoretically allocated to the should-have category was not needed, as the development team was able to complete the must-have items faster than the initial estimate. Thus, this project can be concluded as successful, evidenced by its completion 7 hours ahead of the total estimated time, as shown in Fig. 13.

## Data Availability

All data produced or examined during this study are presented in this paper.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Authors' Contributions

This article is based on a project undertaken by the author. Therefore, the author has implemented Agile-based project development. This article was subsequently written.

## Acknowledgments

The author would like to express his gratitude to Anwar Medika Corporate doctor Nungky Taniasari MARS, Martina Kurnia Rohmah as the rector of Anwar Medika University, and Muhammad Shodikin.

## References

Agustini, F. (2023). Implementasi Metode Scrum Pada Aplikasi Penjualan Peta Dan Buku (Studi Kasus Pada CV Ubo Rampe Palwoko). *Artikel Ilmiah Sistem Informasi Akuntansi*, *3*(1), 36–41. https://doi.org/10.31294/akasia.v3i1.1900

Alshammari, F. H. (2022). Analytical Evaluation of SOA and SCRUM Business Process Management Approaches for IoT-Based Services Development. *Scientific Programming*, *2022*, 1–14. https://doi.org/10.1155/2022/3556809

Armstrong, P. I., Hubert, L., & Rounds, J. (2003). Circular unidimensional scaling: A new look at group differences in interest structure. *Journal of Counseling Psychology*, *50*(3), 297–308. https://doi.org/10.1037/0022-0167.50.3.297

Cohn, Mike. (2011). *Succeeding with agile : software development using Scrum*. Addison-Wesley.

Del Sagrado, J., & Del Águila, I. M. (2021). Assisted requirements selection by clustering. *Requirements Engineering*, *26*(2), 167–184. https://doi.org/10.1007/s00766-020-00341-1

Farid, M. T., Adi Nugroho, H., & Hidayah, I. (2021). Sistem Rekomendasi Minat Anak Menggunakan Alat Tes Rothwell Miller Interest Blank Berbasis Web. *Jurnal Ilmiah ILKOMINFO - Ilmu Komputer & Informatika*, *4*(1). https://doi.org/10.47324/ilkominfo.v4i1.116

Hermans, K. (2023). *Mastering Agile: A Comprehensive Guide to Learn Agile*. Cybellium Ltd.

Istiqomah, D. A., Vikky Aprelia Windarni, Surya Tri Atmaja Ramadhani, & Zein Nur Ichsan. (2024). Peningkatan Pengetahuan Agile Scrum dalam Pengembangan Perangkat Lunak pada Startup Kala Kreatif Indonesia. *JURPIKAT (Jurnal Pengabdian Kepada Masyarakat)*, *5*(3), 915–924. https://doi.org/10.37339/jurpikat.v5i3.1777

Lee, W.-T., & Chen, C.-H. (2023). Agile Software Development and Reuse Approach with Scrum and Software Product Line Engineering. *Electronics*, *12*(15), 3291. https://doi.org/10.3390/electronics12153291

Miranda, E. (2022). *Moscow Rules: A Quantitative Exposé* (pp. 19–34). https://doi.org/10.1007/978-3-031-08169-9_2

Nugraha, D., Nur, I. L., Hidayatuloh, M. T., Laluma, R. H., & Gunawan. (2023). Implementasi Sistem Informasi Manajemen Kantor Menggunakan Scrum Framework Di Desa Wangunsari. *Jurnal Ilmiah Media Sisfo*, *17*(1), 116–124. https://doi.org/10.33998/mediasisfo.2023.17.1.740

Pramudawardani, N. K. P., & Adiati, R. P. (2024). Holland's RIASEC Model: Career and Job Development Assessment for Personnel Administration Staff. *Psikostudia : Jurnal Psikologi*, *13*(3), 361. https://doi.org/10.30872/psikostudia.v13i3.14915

Pratama, S., Ibrahim, S., & Reybaharsyah, M. A. (2022). Jurnal Penggunaan Metode Scrum Dalam Membentuk Sistem Informasi Penyimpanan Gudang Berbasis Web. *INTECH*, *3*(1), 27–35. https://doi.org/10.54895/intech.v3i1.1192

Rad, N. K. . (2021). *Agile Scrum Foundation Courseware English: Based on the 3rd Edition of the Agile Scrum Handbook*. Van Haren Publishing.

Rodriguez, G., Soria, A., & Campo, M. (2016). Measuring the Impact of Agile Coaching on Students' Performance. *IEEE Transactions on Education*, *59*(3), 202–209. https://doi.org/10.1109/TE.2015.2506624

Setiawan, R., Mulyani, A., Ramdan, G. M., & Roji, F. F. (2024). Aplikasi Resep Makanan Bergizi Membantu Pencegahan Stunting Menggunakan Metodologi Agile Framework Scrum. *Jurnal*

*Teknologi Informasi Dan Ilmu Komputer*, *11*(4), 761–770. https://doi.org/10.25126/jtiik.1148384

Suwarno, S., & Jaya, W. S. (2022). Design and Development of Software Project Management System using Scrum. *Journal of Informatics and Telecommunication Engineering*, *5*(2), 483–493. https://doi.org/10.31289/jite.v5i2.6412

Tasrif, E. (2022). RIASEC Holland's reliability and validity on personality of informatics engineering education students in higher education. *JPPI (Jurnal Penelitian Pendidikan Indonesia)*, *8*(1), 11. https://doi.org/10.29210/020221602

**Warna Agung Cahyono** is currently a lecturer in the software engineering study program at Anwar Medika University. He is currently active in application development, particularly web-based ones. His research interests include software development, particularly web-based ones.