Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.079379

ARTICLE

Codenote: Leveraging AI-Driven Personality Grouping to Foster Students' Coding Self-Efficacy

Jia-Rou Lin[1], Chun-Hsiung Tseng[1,*], Hao-Chiang Koong Lin[2] and Andrew Chih-Wei Huang[3]

[1]Department of Electrical Engineering, Yuan Ze University, Taoyuan, Taiwan
[2]Department of Information and Learning Technology, National University of Tainan, Tainan, Taiwan
[3]Department of Psychology, Fo Guang University, Yilan, Taiwan

*Corresponding Author: Chun-Hsiung Tseng. Email: lendle@saturn.yzu.edu.tw

Received: 20 January 2026; Accepted: 12 March 2026; Published: 08 May 2026

ABSTRACT: Effective pair programming relies heavily on optimal partner compatibility, a requirement that is often difficult to scale manually in software engineering education. This study presents the empirical validation of Codenote, an AI-driven Integrated Development Environment (IDE) designed to automate personality-aware group formation. By integrating a behavioral analysis mechanism, Codenote infers student personality traits from coding patterns to construct complementary pairs, thereby facilitating intelligent collaborative learning. To validate the system's effectiveness, a controlled experiment was conducted to assess the impact of this AI-mediated pairing strategy on students' self-efficacy across adaptive, innovative, and persuasive domains. Results indicate that students paired via Codenote's complementary algorithm demonstrated significantly higher adaptive self-efficacy compared to the control group. This suggests that the system's ability to expose learners to diverse problem-solving perspectives effectively enhances their confidence in managing complex and dynamic programming tasks. While no significant improvements were observed in innovative or persuasive self-efficacy, these findings identify specific directions for future system iterations, such as integrating automated scaffolding for creative exploration and leadership. Overall, this study demonstrates the viability of Codenote as an intelligent tool for scaling personalized instruction and highlights the crucial role of AI-driven pairing strategies in fostering psychological readiness for collaborative problem solving.

KEYWORDS: Codenote; pair programming; personality-aware grouping; self-efficacy

1 Introduction

In recent years, programming education has become an essential component of modern curricula due to the increasing importance of computational thinking and problem-solving skills. Programming languages are not only tools for human computer interaction but also form the foundation of systematic and logical reasoning. In Taiwan, educational reforms such as the 12-Year Curriculum Guidelines explicitly emphasize computational thinking and problem-solving abilities to prepare students for real-world challenges [1]. Integrating computational thinking into curriculum design has been shown to enhance students' problem-solving capabilities and deepen their understanding of programming concepts [2]. Despite these advancements, programming remains a highly abstract and challenging subject for most students. Effective learning requires the simultaneous mastery of syntax memorization, comprehension of example code, completion of practice tasks, and interpretation of immediate feedback from development environments [3]. In large-class settings, students who encounter difficulties without timely support often fall behind, creating a negative cycle of frustration, disengagement, and reduced learning outcomes. To address these challenges,

instructional strategies that promote hands-on, collaborative learning have received increasing attention. To address the growing demand for adaptive learning environments, we developed Codenote, which is a personality-aware Integrated Development Environment (IDE) designed with the functionality of AI-powered pair programming.

Pair programming has emerged as a prominent approach for supporting collaborative learning in programming. Originally developed as an agile software development practice, pair programming involves two programmers working together on the same computer to improve code quality and reduce errors. In educational contexts, studies have shown that pair programming reduces students' frustration and cognitive load while also alleviating instructors' teaching burdens, because students no longer rely solely on instructors as their source of guidance [4,5]. However, challenges such as inappropriate pairing, unequal participation, task difficulty mismatches, and classroom constraints may limit its effectiveness [6]. Research also suggests that pairing students based on complementary personality traits can enhance learning outcomes, particularly in remote collaboration contexts [7].

A major limitation in implementing personality-informed strategies lies in the reliance on traditional self-report questionnaires, which are often time-consuming and prone to low-quality or inattentive responses. To address this limitation, the present study proposes inferring personality traits from students' operational behaviors and note-taking activities. Prior studies have suggested that machine learning approaches can be applied to analyze note-taking patterns in order to infer individual personality traits [8]. In addition, Meidenbauer et al. and Buker et al. examined mouse movement patterns and keystroke dynamics as behavioral indicators of personality. Their findings indicate that features such as mouse pauses, clicks, and typing dynamics are significantly associated with traits including Conscientiousness and Agreeableness [9]. Given that programming practice inherently involves frequent note-taking and code annotation, these observable behaviors provide a practical and unobtrusive source of data for inferring students' personality traits.

Based on the aforementioned research insights, this study develops a personality trait based pair programming framework supported by a pair programming assistant system to implement a one-to-many pairing model in classroom settings. Students are grouped according to inferred personality traits and engage in collaborative learning with the support of the assistant software. This design aims to enhance the scalability of instructional practices while preserving the advantages of pair programming in promoting learning interaction and conceptual understanding. The primary objective of this study is to examine the impact of a personality-informed, assistant-supported pair programming model on students' programming competence, thereby providing empirical evidence for the development of more adaptive and effective programming learning environments. In addition, this study adopts self-efficacy as a core validation indicator to investigate whether personality-based pair programming can effectively enhance programming learning outcomes. Prior research has indicated a significant positive correlation between self-efficacy and academic achievement in programming courses. Martínez-Mejía and Rodríguez-Villanueva further found that self-efficacy and learning motivation have positive effects on academic performance and contribute to the development of problem-solving skills and creativity among programming learners [10]. To empirically validate these effects within an intelligent environment, the present study implements the proposed framework using the Codenote system.

The architectural framework and core functional design of the Codenote system were initially conceptualized and presented in our preliminary work at TCSE 2025[1]. As detailed in that study, traditional Integrated Development Environments (IDEs) often lack mechanisms to capture the psychological nuances of learners'

[1] https://tcse2025.seat.org.tw

behaviors. To address this, the original Codenote prototype was developed to demonstrate how students' operational behaviors—such as note-taking frequency and code editing patterns—could be utilized to infer personality traits without invasive questionnaires. While the TCSE 2025 study focused on the technical feasibility of the behavior-to-personality inference algorithm, the present study extends this foundation by deploying the system in a live educational setting. Specifically, we upgraded the system to include an active intervention module: the AI-powered pair programming engine. This evolution marks the transition of Codenote from a passive assessment tool to an intelligent intervention system, aiming to empirically validate its impact on student learning outcomes.

2 Theoretical Background

2.1 Personality Traits and Their Relationship with Learning

The relationship between personality traits and academic achievement has long attracted scholarly attention. Prior research indicates that higher levels of conscientiousness are associated with effective time management, diligence, and attention to detail, all of which contribute to improved academic outcomes [11]. Additionally, openness to experience, characterized by a willingness to engage with new ideas and experiences, has been shown to be positively related to academic performance, particularly during the early stages of education [12,13].

Based on this body of research, incorporating personality traits as parameters within digital learning systems offers a promising approach for adaptive instructional design. By enabling systems to automatically adjust instructional strategies and pairing learners with complementary personality profiles for pair programming activities, it becomes possible to leverage individual differences to achieve more effective learning outcomes. Building on this premise, the present study employs Codenote to operationalize these theoretical insights. Unlike traditional approaches that rely on static assessments, Codenote utilizes real-time behavioral data to dynamically infer these critical traits. This capability enables the system to implement an automated, scalable mechanism for forming complementary groups, specifically designed to foster the psychological readiness required to enhance students' coding self-efficacy.

2.2 Pair Programming

Pair programming was originally developed as an agile software development practice in the software industry, in which two programmers work collaboratively on the same computer to improve design quality and reduce software defects. Ainsworth and Th argue that the effectiveness of pair programming in software development can be attributed to clear role differentiation and the facilitation of discussion between partners, which help reduce cognitive load and thereby alleviate the burden on working memory [14].

However, the effects of pair programming are not uniformly positive. In recent years, pair programming has been increasingly adopted in programming courses, yet its instructional effectiveness remains a subject of debate. McDowell et al. conducted a large-scale experiment involving 404 students at the University of California and found that, in a CS1 course, students who engaged in randomly assigned pair programming were more likely to persist until the final examination compared with those who did not use pair programming (90.8% vs. 80.4%). Nevertheless, no significant differences were observed in final course grades between the two groups [15]. McChesney conducted a three-year longitudinal study and reported that, on average, pair programming had a positive effect on students'learning outcomes; however, the results were inconsistent across different years. Furthermore, qualitative findings revealed that some students perceived pair programming as imposing considerable learning pressure and reported relatively low productivity

during the process [16]. Notably, 29% of the participants in this study still considered discussion during pair programming to be an important component of the learning experience.

Further evidence suggests that algorithmic team formation strategies play a critical role in determining the effectiveness of pair programming. In the context of group-aware learning analytics, Poonam and Yasser reported that pairing students with different personality traits through remote collaboration resulted in the most favorable learning outcomes [7]. Similarly, Demir and Seferoglu validated this finding and further examined grouping strategies based on multiple learner characteristics. Their results indicated that pairing students with similar learning styles and levels of interpersonal closeness led to more enjoyable pair programming experiences, whereas pairing students with differing levels of interpersonal closeness and self-regulation resulted in better learning performance [17]. These findings suggest that, in both industrial and educational contexts, pair programming does not inherently guarantee improved work or learning outcomes, and that careful consideration of partner characteristics is essential.

In summary, the existing literature indicates that pair programming based on random assignment or solely on ability measures, such as programming skills or prior academic performance, does not yield consistently positive outcomes. However, several common principles have emerged that may enhance the effectiveness of pair programming. These include pairing learners with complementary characteristics, assigning distinct roles within the pair, and encouraging active discussion between partners to reduce cognitive load. Collectively, these principles provide important theoretical and practical foundations for the design of more effective pair programming instructional models. The Codenote system proposed in this study directly translates these principles into practice through its AI-driven pairing engine. By ensuring that students are matched based on the complementary traits identified in the literature rather than random assignment, the system seeks to maximize the benefits of collaborative interaction. This study aims to provide empirical evidence on how such structured, technology-mediated pairing strategies specifically influence students' self-efficacy in a real-world educational setting.

2.3 Self-Efficacy and Programming

Self-efficacy is widely regarded as a critical psychological factor reflecting students' ability to perform effectively within the domain of programming. Kovari and Katona characterize programming self-efficacy not merely as a skill metric, but as a composite construct that combines independent problem-solving abilities, self-confidence, and the sustained motivation required for coding tasks [18]. Students who demonstrate higher levels of this trait tend to comprehend abstract program concepts more effectively. Empirical research supports this link; for instance, Avcu and Ayverdi discovered that among gifted students, a significant positive correlation exists between programming self-efficacy and computational thinking skills [19]. To mitigate these challenges, Schultz and Blaszczyk concluded that the use of process guides can reduce negative emotional experiences, thereby increasing self-efficacy and lowering dropout rates [20]. Additionally, El Khoury et al. highlighted that robust self-efficacy is essential for overcoming initial hurdles and achieving long-term success in programming courses [21].

In the context of this study, self-efficacy is adopted as a primary metric for validating the Codenote system. By leveraging AI-driven pairing to create supportive, complementary partnerships, the system aims to reduce the "negative emotional experiences" and "initial challenges" highlighted in the literature, thereby fostering a learning environment conducive to the development of robust programming self-efficacy. Specifically, students' self-efficacy was assessed using the self-efficacy scale developed by Ng and Lucianetti [22], a widely recognized and highly cited instrument in the field. The scale further divides self-efficacy into innovative self-efficacy, persuasion self-efficacy, and adaptive self-efficacy. According to Ng and Lucianetti, the scale measures participants' innovative behaviors. Innovative self-efficacy is the self-view that

one has the ability to produce novel ideas. Persuasion self-efficacy is the extent to which employees are confident in their ability to convince others to accept and adopt new ideas. Adaptive self-efficacy refers to an individual's perceived ability to handle change in the workplace.

3 Methodology

This section details the research design, system implementation, and experimental procedures conducted to evaluate the pedagogical impact of the Codenote system. We first describe the cloud-based system architecture developed to facilitate real-time classroom interaction (Section 3.1). Subsequently, we detail the AI-driven classification models and feature engineering processes used to enable the heterogeneous pairing mechanism (Section 3.2). Finally, the experimental setup, including the participant demographics, learning activities, and evaluation metrics, is presented to outline how the intervention was empirically validated in a live educational setting.

3.1 System Architecture and Implementation

To facilitate scalable classroom deployment and robust real-time interaction, Codenote is architected as a cloud-based Web service adhering to a Service-Oriented Architecture (SOA). The system is structurally divided into two primary layers: a server-side backend handling core logic and data persistence, and a client-side frontend managing user interaction and feature extensions. The system architecture of Codenote is shown in Fig. 1.

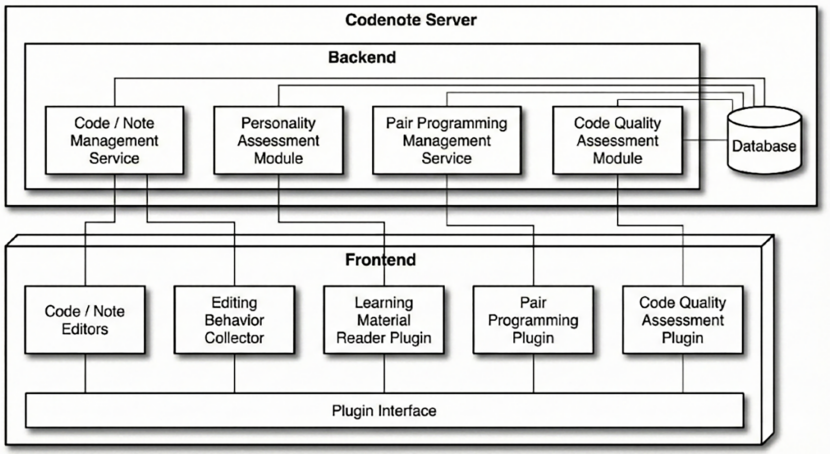

Figure 1: System architecture of Codenote.

The backend, positioned at the upper layer of the architecture, is implemented using the SpringBoot framework. It comprises four distinct micro-services arranged to handle specific functional domains. From left to right, these components are: the *Code/Note Management Service*, the *Personality Assessment Module*, the *Pair Programming Management Service*, and the *Code Quality Assessment Module*. Underpinning these services is a centralized Database, which maintains connectivity with all aforementioned modules to ensure data consistency and accessibility across the platform.

The frontend, located at the lower layer, is built upon modern JavaScript modules and centers on user interaction. Its core components include the *Code/Note Editors* (powered by the Monaco Editor) and the *Editing Behavior Collector*. Surrounding these are specialized plugins designed for educational interventions: the *Learning Material Reader Plugin*, the *Pair Programming Plugin*, and the *Code Quality Assessment Plugin*.

A dedicated Plugin Interface operates at the foundation, connecting these three plugins directly to the core editing environment, allowing for dynamic feature injection.

The architecture establishes precise communication channels between frontend components and backend services to facilitate data exchange:

- Content Management: Both the *Code/Note Editors* and the *Learning Material Reader Plugin* interface directly with the backend *Code/Note Management Service* to handle file operations and learning material retrieval.
- Behavior Analysis: The *Editing Behavior Collector* transmits user operational data to the backend *Personality Assessment Module* for personality trait analysis.
- Collaboration: The frontend *Pair Programming Plugin* maintains a direct connection with the backend *Pair Programming Management Service* to synchronize coding sessions.
- Evaluation: The *Code Quality Assessment Plugin* communicates directly with the backend *Code Quality Assessment Module* to facilitate instant feedback on code performance.

To clarify how the Codenote system integrates into the instructional design, Fig. 2 below illustrates the end-to-end operational workflow.

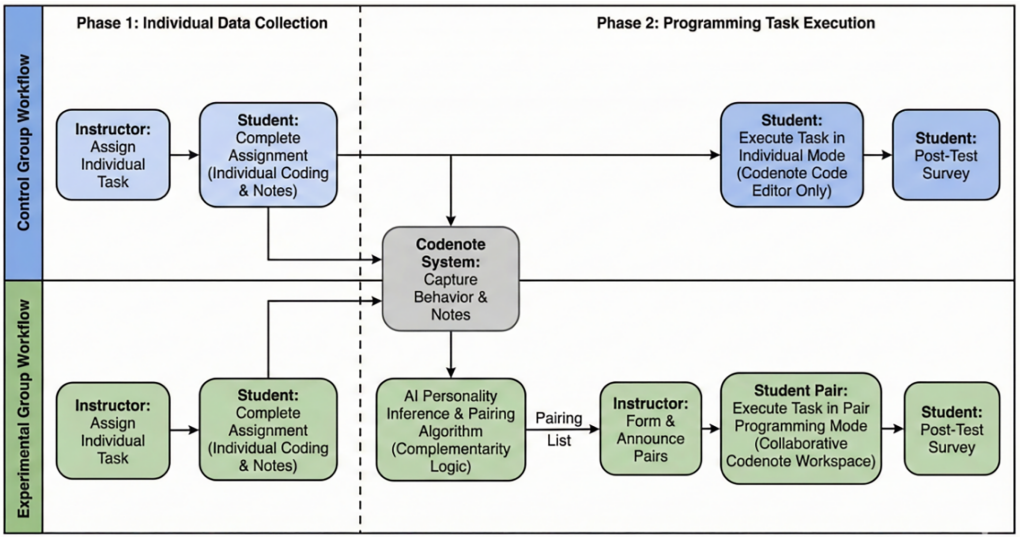

Figure 2: Experiment flow.

The process is divided into two phases:

1. Phase 1: Individual Data Collection (Baseline): All students initially engage in individual coding tasks. The system captures their operational behaviors and note-taking patterns (Input) to train the personality inference model, which outputs a specific personality profile for each learner.
2. Phase 2: Intervention (Control vs. Experimental):
   (a) Control Group: Students continue to execute programming tasks individually using the standard Codenote editor.
   (b) Experimental Group: The system's pairing algorithm utilizes the inferred personality profiles to generate a Pairing List based on complementarity. Students then enter the Collaborative Workspace, where they assume Driver/Navigator roles supported by the system.

3.2 Personality-Based Grouping Mechanism

Previous studies have indicated that using a single personality trait is insufficient to fully explain the effectiveness of pair programming on learning outcomes [23,24]. The present study adopts a personality-based grouping approach, considering multiple traits as key parameters for pair programming assignments. As prior research has demonstrated a significant relationship between programming behaviors and personality traits [25], this study employs students' actual programming and operational behaviors as the basis for personality trait classification.

To operationalize this behavioral analysis, the system employs a specialized data structure named *CodeDiffBean* to capture granular user interactions. The *CodeDiffBean* records two primary attributes: *diffString*, which captures the content of user inputs, and *diffSource*, which identifies whether the input originated from the source code editor or the note-taking panel. An *Editing Behavior Collector* module runs in the background, monitoring the frequency of context switching between coding and note-taking, as well as the volume of typing activity.

The personality trait classification model employed in this study was pre-trained and rigorously validated during a preliminary pilot phase of this research project. The training dataset for this initial phase consisted of 45 undergraduate students at Yuan Ze University participating in a frontend web programming course. Ground-truth labels for personality traits were established using the Five Factor Personality Traits questionnaire, administered prior to the data collection. The model utilizes a two-layer Random Forest architecture to process students' note-taking content and system operation behaviors (e.g., switching frequency between code and note interfaces). While the foundational validation of the AI grouping mechanism was established during the aforementioned pilot phase, the selection of the two-layer Random Forest architecture was driven by the specific nature of the behavioral dataset. First, given the relatively small training sample (N = 45), an ensemble method like Random Forest was chosen over a single decision tree to mitigate the risk of overfitting through bagging. Second, human behavioral data, such as the variance in coding transitions and keyword frequencies, often exhibits complex, non-linear interactions. Random Forest natively captures these non-linearities more effectively than simpler linear models (e.g., Logistic Regression or Support Vector Machines) without requiring strict assumptions regarding data distribution. Finally, the two-layer hierarchical architecture was implemented to optimize multi-class classification; by decomposing the task into sequential binary decisions, the system more stably isolated distinct personality profiles within the constrained dataset. The predictive models were implemented using the randomForest package in R. To ensure strict reproducibility, a fixed random seed was applied (set. seed(5)). For the first-layer model, the number of trees (ntree) was set to 750, and the number of variables sampled at each split (mtry) was 1. For the second-layer model, ntree was set to 500 with an mtry of 3. Given the relatively small size of the pre-training dataset (N = 45), we deliberately avoided aggressive hyperparameter tuning (such as deep grid search or cross-validation matrices) to prevent over-optimizing to the sample. Instead, ntree and mtry were determined empirically based on the stabilization of the Out-of-Bag (OOB) error. Other architectural parameters including maximum tree depth, feature subsampling (standard bootstrap with replacement), class weighting, and minimum samples per leaf (default = 1) were kept at their robust library defaults. This conservative configuration strategy was explicitly chosen to minimize variance and control the risk of overfitting.

To construct the predictive models, the raw behavioral log data which contained no missing values after the initial extraction phase was engineered into three distinct feature categories. First, semantic features included the frequencies of specific note-taking keywords (e.g., questions or other annotations), which were standardized using Z-score scaling. Second, temporal features were constructed by dividing the total coding session into five equal time segments. To account for varying session lengths, these segments were

normalized as ratios of the total timestamp. Third, behavioral variance metrics were calculated to capture the fluctuation in students' editor-switching behaviors. To ensure model interpretability and avoid the "black-box" nature of automated dimensionality reduction, feature selection was conducted using the Boruta algorithm. As a wrapper method built around Random Forest, Boruta rigorously isolates all statistically relevant variables by comparing original feature importance against randomized shadow features. Guided by these results, the model formulas were optimized to prevent overfitting. The first-layer classification utilized a targeted subset of critical semantic features confirmed as highly relevant, while the second-layer model incorporated a broader combination of temporal and variance features. This transparent, Boruta-driven feature engineering process ensured that the AI mechanism relied on meaningful, pedagogical behavioral patterns rather than noise.

Due to the constrained size of the pre-training dataset (N = 45), traditional techniques such as k-fold cross-validation or hold-out validation would significantly reduce the available training data, potentially destabilizing the model. Therefore, performance was evaluated using Out-of-Bag (OOB) estimation. OOB error is widely recognized as an unbiased and highly reliable estimate of true test set error in Random Forests, particularly suited for small sample sizes. The first layer classifies students based on note-taking features to identify high conscientiousness and low openness traits. This layer achieved an Out-of-Bag (OOB) error rate of 24.32% with a significance level of $p = 0.0185$. The second layer further classifies the remaining students using vectors derived from both notes and behavioral actions, achieving a lower OOB error rate of 10.71%. To provide a comprehensive evaluation beyond the raw OOB error rate, we calculated standard classification metrics (Accuracy, Precision, Recall, and Macro F1-score) based on the OOB confusion matrices.

The first-layer classification model achieved an overall accuracy of 75.68%, with a Precision of 0.769, Recall of 0.625, and a balanced Macro F1-score of 0.745. The second-layer model, distinguishing between the remaining two profiles, demonstrated even higher robustness, achieving an accuracy of 85.71%. Notably, it yielded a Precision of 0.778 and a Recall of 1.00 for the target group, culminating in a Macro F1-score of 0.854. These metrics confirm that the models maintain a strong balance between precision and recall, ensuring the reliability of the automated personality grouping mechanism prior to the educational intervention. This two-stage approach allows Codenote to infer personality traits from non-invasive behavioral data with established empirical validity, enabling the automated grouping strategy used in the current experimental intervention. Fig. 3 shows the confusion matrices:

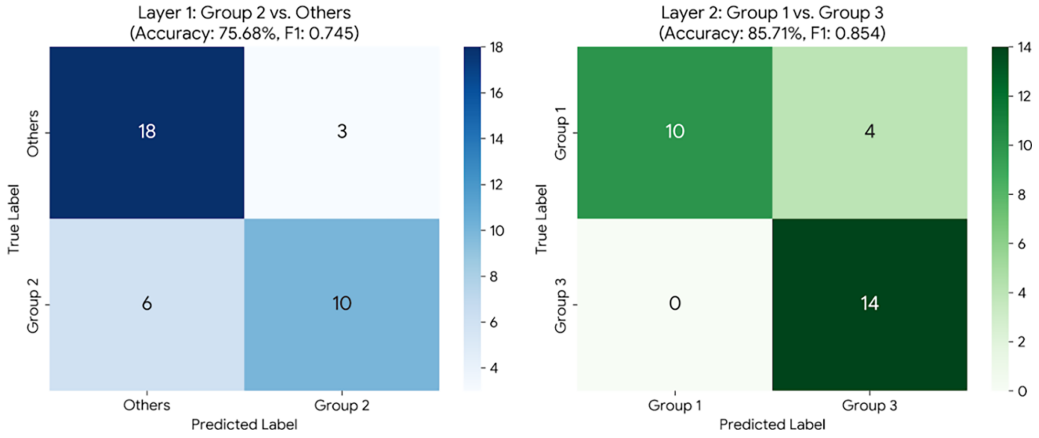

Figure 3: The confusion matrix.

Specifically, we utilized the Codenote system to collect data on students'note-taking and operational behaviors in a front-end web design course, and subsequently classified students according to personality traits such as conscientiousness and openness to experience, which have been shown to be closely associated with effective learning strategies. In the present study, personality trait assessment was conducted prior to the instructional intervention and remained fixed throughout the experimental period; subsequently; students with different personality traits were paired together for programming activities.

To ensure the classification thresholds were ecologically valid for the specific student demographic, we utilized a cumulative dataset collected from 165 undergraduate students at Yuan Ze University between 2021 and the onset of this experiment. Ground-truth personality traits were measured using the International Personality Item Pool (IPIP) scale. The thresholds for determining 'High' vs. 'Low' levels for each trait were established using a median-split method based on this local historical dataset. Consequently, the pairing algorithm defined a student as 'High Conscientiousness' if their inferred score exceeded the historical median of this local population, ensuring that the grouping criteria were calibrated to the specific characteristics of the student cohort.

While calculating the theoretical end-to-end exact-match accuracy of the cascade pipeline yields approximately 64.87% (0.7568×0.8571), this metric must be interpreted within the pedagogical context of the system. The fundamental objective of the Codenote system is not to serve as a perfect psychometric diagnostic tool, but to automate heterogeneous pairing, i.e., grouping students with divergent traits to foster collaborative problem-solving. In practical terms, achieving this pedagogical goal does not require flawless fine-grained classification. The robust binary separation executed by the first-layer model isolates the most distinct personality profile from the rest, while the second layer ensures behavioral contrast among the remaining students. Therefore, even if minor misclassifications occur due to algorithmic error propagation, the baseline complementary difference required for paired students is mathematically and practically preserved. By functioning as an unobtrusive automated proxy for traditional questionnaires (thereby eliminating survey fatigue), the AI mechanism successfully generated the "productive friction" necessary for learning. The significant improvements in students' self-efficacy observed in this study empirically validate that this algorithmic accuracy baseline is highly sufficient for real-world educational interventions.

3.3 Runtime Code Analysis and Feedback Mechanism

Beyond personality assessment, Codenote incorporates a real-time Code Quality Assessment Module to support student autonomy during pair programming. Unlike static code analysis tools, our system implements a dynamic runtime evaluation. When a student edits a project (HTML/JavaScript), the system instantiates an invisible iframe within the browser context. This isolated environment executes the student's code in the background, intercepting runtime errors and console outputs. The system then compares these runtime events against pre-configured validation scripts defined by the instructor. This mechanism allows the system to simulate a "developer console" experience, identifying logic errors that syntax highlighters miss, and providing immediate, context-aware feedback to the pair, thereby reducing the cognitive load on the navigator.

To ensure experimental validity and isolate the pairing strategy as the sole independent variable, this runtime analysis functionality was uniformly activated across all conditions. Specifically, the feature was enabled for the 'Driver' (code writer) in the pair programming groups, providing them with immediate diagnostic feedback during collaboration. Simultaneously, students in the control group, who worked individually, also had full access to this runtime analysis feature. This design ensures that any observed differences in learning outcomes are attributable to the personality-based pairing intervention rather than disparities in tool functionality or debugging support.

3.4 Experimental Procedure

The experimental design of this study consisted of two stages. In the first stage, a pretest was administered to assess students' programming competence and self-efficacy levels. Specifically, self-efficacy was measured using the multi-dimensional scale developed by Ng and Lucianetti [22]. In this study, these dimensions were operationalized to assess students' confidence in generating novel coding solutions (innovative), influencing peer decisions (persuasive), and coping with debugging challenges (adaptive) within the classroom context. In the second stage, students were assigned to either an experimental or a control group, ensuring no significant initial differences in self-efficacy existed between them. Participants in the experimental group engaged in programming practice using the pair programming assistant component of the Codenote system. They were paired based on personality trait groups, with each pair comprising a navigator and a driver. Based on pretest results, higher-competence students were designated as navigators, while those with lower competence acted as drivers; these roles remained constant throughout the intervention. Conversely, students in the control group practiced with the same system, but without the pair programming function. Finally, posttest self-efficacy levels were compared to evaluate the effects of the proposed approach. This design ensured that any observed differences could be attributed to the pairing strategy rather than system usage.

To facilitate smooth and efficient pair programming, this study developed a dedicated pair programming module integrated into the Codenote system[2]. As implemented in our previous work, the system is built as a Web service using the SpringBoot framework, featuring a backend *Personality Assessment Module* that utilizes a Random Forest model to process student behaviors. Before class, teachers can pre-assign student groups within the system. By integrating this AI-driven personality trait assessment model, the system allows teachers to review the distribution of students' personality traits and to form groups accordingly. The user interface for personality assessment is shown in Fig. 4.

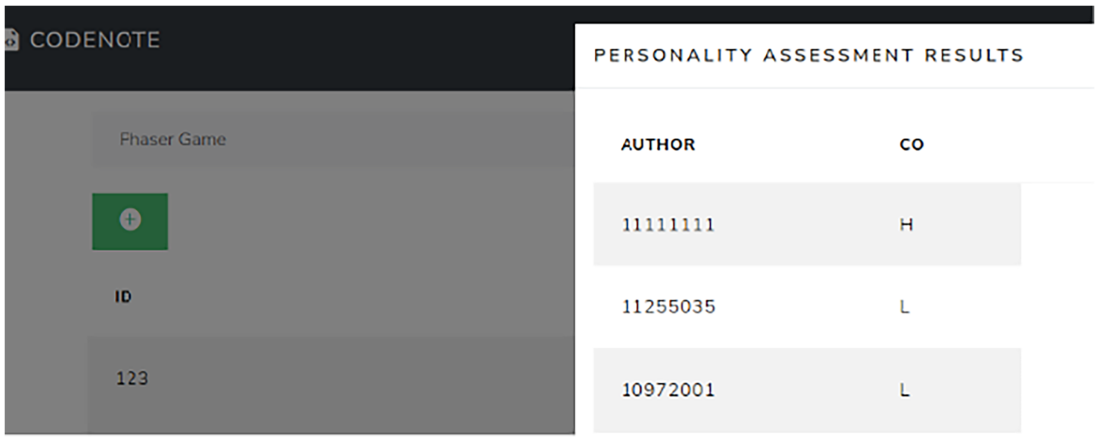

Figure 4: Codenote personality trait assessment interface.

During class, once students log into the system, pairing is automatically performed based on these predefined group assignments. Within each pair, one student is designated as the code writer (driver), while the other serves as the mentor (navigator). The mentor can observe their partner's code in real time, and both participants are able to communicate synchronously through the system's built-in messaging interface. Through this role structure and communication mechanism, the system is designed to promote sustained

[2] https://gitlab.com/zoe841228/codenote

interaction and timely feedback, thereby supporting effective collaborative learning. The system interface for paired programming is shown in Fig. 5.

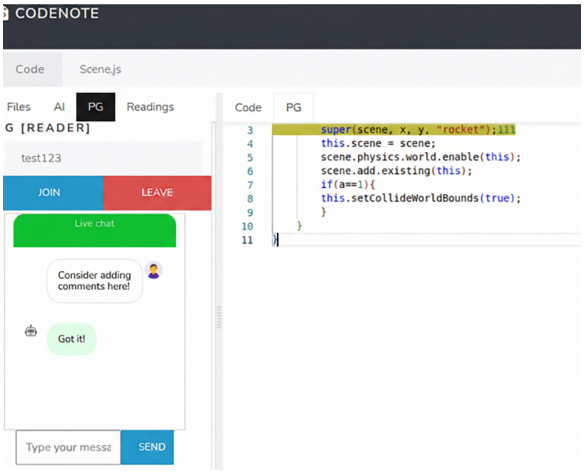

Figure 5: Pair programming interface.

4 Results

During the semester, a total of 30 students from Yuan Ze University (Taoyuan, Taiwan) were recruited to participate in the experiment and were randomly assigned to either a control group or an experimental group, with 15 students in each group. Students in the control group used the same programming editor for learning activities but did not participate in pair programming. In contrast, students in the experimental group used the same programming editor with the pair programming function enabled throughout the learning activities.

During the implementation of pair programming, students in the experimental group were paired according to the results generated by the personality trait analysis module. The grouping process was based on students' levels of Conscientiousness and Openness to Experience, two dimensions of the Big Five personality traits, with high and low trait levels used to determine whether students' personality profiles were complementary. Based on this classification, individuals with complementary personality traits were paired together. As the total number of participants was an odd number, one pair programming group included a teaching assistant in order to maintain consistency in the experimental design.

Students' self-efficacy was assessed using pretest and posttest measures, employing the self-efficacy scale developed by Ng and Lucianetti [22], which comprises three dimensions: adaptive self-efficacy, innovative self-efficacy, and persuasive self-efficacy. Table 1 shows the descriptive results.

The analysis results indicated that there was no significant difference between the control group and the experimental group in adaptive self-efficacy at the pretest stage. However, after the completion of the experiment, a significant difference in adaptive self-efficacy was observed between the two groups, as illustrated in Fig. 6.

Table 1: Descriptive statistics.

		Valid	Missing	Mean	Std. Deviation	Minimum	Maximum
Innovative Self-Efficacy (Pre-test)	Control	15	0	13.47	3.25	7.00	21.00
	Experimental	15	0	12.80	2.81	8.00	18.00
Persuasive Self-Efficacy (Pre-test)	Control	15	0	22.73	5.48	15.00	35.00
	Experimental	15	0	21.27	5.35	12.00	31.00
Adaptive Self-Efficacy (Pre-test)	Control	15	0	20.73	5.50	14.00	32.00
	Experimental	15	0	20.27	6.27	10.00	30.00
Innovative Self-Efficacy (Post-test)	Control	15	0	12.73	2.94	6.00	18.00
	Experimental	15	0	13.47	3.11	6.00	18.00
Persuasive Self-Efficacy (Post-test)	Control	15	0	21.33	5.53	12.00	34.00
	Experimental	15	0	21.20	4.86	13.00	30.00
Adaptive Self-Efficacy (Post-test)	Control	15	0	20.07	4.80	11.00	29.00
	Experimental	15	0	30.33	7.08	14.00	42.00

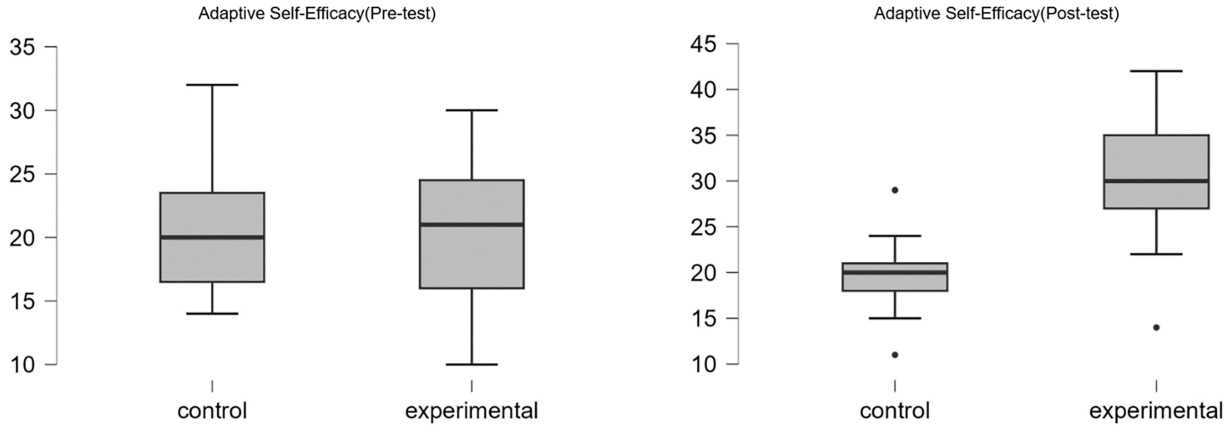

Figure 6: Adaptive self-efficacy: pre-test and post-test comparison between groups.

Prior to conducting the inferential analysis, the assumptions of normality and homogeneity of variance were examined. A Test of Normality using the Shapiro–Wilk method was performed for both groups, and the results showed that all p-values were greater than 0.05, indicating that the data for each group followed a normal distribution, as shown in Table 2.

Table 2: Test of normality (Shapiro-Wilk).

		W	*p*
Innovative Self-Efficacy (Pre-test)	Control	0.954	0.592
	Experimental	0.954	0.591
Persuasive Self-Efficacy (Pre-test)	Control	0.931	0.284
	Experimental	0.951	0.547

(Continued)

Table 2 (continued)

		W	*p*
Adaptive Self-Efficacy (Pre-test)	Control	0.912	0.143
	Experimental	0.953	0.566
Innovative Self-Efficacy (Post-test)	Control	0.864	0.027
	Experimental	0.942	0.406
Persuasive Self-Efficacy (Post-test)	Control	0.933	0.306
	Experimental	0.944	0.435
Adaptive Self-Efficacy (Post-test)	Control	0.918	0.179
	Experimental	0.960	0.699

In addition, a Test of Equality of Variances using Levene's test was conducted to assess the homogeneity of variances between the two groups. The results indicated that all p-values were also greater than 0.05, suggesting that the assumption of equal variances was satisfied, as presented in Table 3.

Table 3: Test of equality of variances (Levene's).

	F	df_1	Df_2	*p*
Innovative Self-Efficacy (Pre-test)	0.026	1	28	0.873
Persuasive Self-Efficacy (Pre-test)	0.079	1	28	0.781
Adaptive Self-Efficacy (Pre-test)	0.266	1	28	0.610
Innovative Self-Efficacy (Post-test)	0.207	1	28	0.652
Persuasive Self-Efficacy (Post-test)	0.105	1	28	0.748
Adaptive Self-Efficacy (Post-test)	2.921	1	28	0.099

Based on these results, the data met the assumptions required for parametric testing, and an Independent Samples t-test was subsequently applied. The statistical results are reported in Table 4. The findings reveal that personality trait–based pair programming produced a significant positive effect on the adaptive dimension of self-efficacy. To address the potential inflation of Type I error due to multiple comparisons (three dimensions of self-efficacy), a Bonferroni correction was considered ($\alpha = 0.05/3 = 0.016$). The results of the Independent Samples t-test (Table 4) indicate that the difference in Adaptive Self-Efficacy remained statistically significant even under this stricter threshold ($p < 0.001$). Furthermore, given the small sample size (N = 30), Hedges' g was reported as the measure of effect size to provide a less biased estimate than Cohen's d. In contrast, no statistically significant differences were found between the experimental and control groups with respect to innovative self-efficacy or persuasive self-efficacy.

Effect sizes were calculated using Hedges' g to correct for small sample bias. Notably, the effect size for adaptive self-efficacy (g = 1.65) indicates a substantial practical impact of the intervention.

Table 4: Independent samples *t*-test.

	Test	Statistic	df	*p*	Effect Size (Hedges' g)	SE Effect Size
Innovative Self-Efficacy (Pre-test)	Welch	0.601	27.426	0.553	0.214	0.357
	Mann-Whitney	127.000		0.557	0.129	0.211
Persuasive Self-Efficacy (Pre-test)	Welch	0.741	27.983	0.465	0.263	0.358
	Mann-Whitney	128.500		0.519	0.142	0.211
Adaptive Self-Efficacy (Pre-test)	Welch	0.217	27.525	0.830	0.077	0.356
	Mann-Whitney	111.000		0.967	−0.013	0.211
Innovative Self-Efficacy (Post-test)	Welch	−0.663	27.907	0.513	−0.236	0.358
	Mann-Whitney	91.500		0.381	−0.187	0.211
Persuasive Self-Efficacy (Post-test)	Welch	0.070	27.549	0.945	0.025	0.355
	Mann-Whitney	116.000		0.901	0.031	0.211
Adaptive Self-Efficacy (Post-test)	Welch	−4.649	24.637	<0.001*	−1.652	0.461
	Mann-Whitney	27.000		<0.001*	−0.760	0.211

Note: *indicates statistical significance at the $p < 0.05$ level.

5 Discussion

5.1 The Role of Personality Complementarity in Pair Programming

The study examined the effects of Codenote-facilitated pair programming based on personality trait complementarity on students' self-efficacy, with particular attention to adaptive, innovative, and persuasive dimensions. Previous research has shown that pair programming can be beneficial in both industrial and educational settings; however, its effectiveness is highly dependent on pairing strategies and the structure of collaboration rather than on the mere implementation of collaborative programming. Empirical studies have consistently indicated that grouping strategies grounded in learner characteristics, such as personality traits, learning styles, interpersonal closeness, and self-regulation, yield more favorable outcomes than random or ability-based pairing. Specifically, Poonam and Yasser reported that pairing students with different personality traits in a remote collaboration context resulted in the most favorable learning outcomes, suggesting that heterogeneity in personality traits may support more effective learning processes [7]. Similarly, Demir and Seferoglu examined grouping strategies based on multiple learner characteristics and found that pairing students with similar learning styles and levels of interpersonal closeness led to more enjoyable pair programming experiences, whereas pairing students with differing levels of interpersonal closeness and self-regulation was associated with better learning performance [17]. Taken together, these findings indicate that different pairing criteria may lead to different types of outcomes, and that complementary learner characteristics are particularly relevant when the instructional focus is on learning performance rather than on subjective experience.

Building on this body of research, the present study adopted Codenote's AI-driven pairing mechanism that grouped students with complementary traits and examined its effects on multiple dimensions of self-efficacy. Results indicated no significant difference in adaptive self-efficacy between the groups at the pretest stage, suggesting they were comparable prior to the intervention. Following the implementation of Codenote-supported pair programming, the experimental group demonstrated significantly higher adaptive

self-efficacy than the control group. This finding aligns with earlier research emphasizing that the effectiveness of pair programming stems from the quality of interaction—including discussion, role differentiation, and mutual monitoring—rather than from collaboration alone. A plausible explanation is that personality-based pairing exposed students to diverse perspectives and coping strategies. As pair programming requires learners to adjust strategies, negotiate solutions, and respond to errors in real time, interacting with partners who approach problems differently may foster flexibility and resilience. Consequently, this enhances students' perceived ability to manage novel or uncertain situations, an interpretation consistent with qualitative findings that highlight interpersonal interaction as a central component of effective pair programming.

5.2 Dimension-Specific Effects on Self-Efficacy

In contrast, no significant effects were observed for innovative self-efficacy or persuasive self-efficacy. This suggests that the AI-based pairing algorithm alone may be insufficient to influence these dimensions. Prior research has indicated that innovative self-efficacy is closely associated with opportunities for creative exploration and open-ended problem solving, while persuasive self-efficacy is more strongly related to leadership roles, communication demands, and explicit decision-making authority. Although system-mediated collaborative programming was implemented in the present study, the instructional design and system features may not have explicitly emphasized these elements, thereby limiting potential gains in these dimensions of self-efficacy. Overall, the findings reinforce the view that pair programming does not inherently produce uniformly positive effects across all psychological or learning outcomes. Instead, its effectiveness appears to be dimension-specific and contingent upon instructional design and tool capability decisions. The significant improvement observed in adaptive self-efficacy suggests that Codenote's intelligent pairing strategy is particularly effective in supporting students' confidence in coping with complex and dynamic programming tasks. By demonstrating a selective effect on adaptive self-efficacy, this study contributes to the literature by clarifying the conditions under which AI-powered collaborative tools may enhance learners' psychological readiness rather than assuming broad benefits across all aspects of self-efficacy.

While significant improvements were not observed in innovative or persuasive domains, the isolated gain in adaptive self-efficacy represents a critical pedagogical milestone in computer science education. The field of programming is characterized by rapid technological evolution, where languages, frameworks, and methodologies are in a constant state of flux. Consequently, the ability to adapt to change is arguably the most vital attribute for long-term success. As recent research on career development highlights, individuals equipped with strong career adaptability are significantly more likely to engage in proactive career behaviors, enabling them to successfully navigate uncertainty and rapid technological shifts [26]. In the context of this study, the significant increase in adaptive self-efficacy indicates that personality-based pair programming successfully cultivated students' psychological readiness to navigate uncertainty. This suggests that students are not merely learning to code, but are developing the 'change agility' required to proactively learn new skills and drive technological innovation in their future workplaces. Therefore, even in the absence of immediate gains in innovation or persuasion, the enhancement of adaptive capabilities provides a foundational scaffold for students' persistent engagement and lifelong learning in engineering disciplines.

5.3 Pedagogical Implications in the Era of Generative AI

In light of the rapid proliferation of Generative AI (GenAI) in programming education, it is pertinent to distinguish the impact of human peer pairing from AI assistance. While GenAI tools can effectively scaffold syntax acquisition and debugging, the significant improvement in adaptive self-efficacy observed in this study likely stems from the socio-cognitive demands of human partnership. Specifically, the need

to negotiate conflicting ideas and navigate interpersonal dynamics. Current AI assistants, which typically function as passive information retrievers, may not replicate this 'productive friction' that fosters psychological resilience.

However, the core principle validated in this study offers a crucial design blueprint for next-generation AI tutors. Rather than providing generic responses, future AI agents could be engineered with specific 'personas' (e.g., a highly conscientious 'Navigator' Agent) to complement the learner's detected traits. By simulating the heterogeneous pairing logic validated in this study, AI-mediated environments could potentially recreate the psychological benefits of human collaboration, moving from simple code generation to personalized socio-emotional scaffolding.

Compared with recent related literature, this focus on "adaptive resilience" offers a distinct pedagogical value. For instance, Yilmaz and Karaoglan Yilmaz [27] demonstrated that Generative AI tools (e.g., ChatGPT) significantly boost students' programming self-efficacy by assisting with task completion and debugging. Similarly, a recent systematic review by Massaty et al. [28] concluded that AI-driven tools generally enhance self-efficacy by providing "tailored feedback and support," which directly reinforces students' confidence in their academic abilities. While these approaches effectively build confidence through task success (i.e., "I can solve this because AI helped me"), our personality-based pairing strategy fosters confidence through adaptability. By enhancing adaptive self-efficacy, our system does not merely make programming easier; it equips students with the change agility required to face uncertainty. This resonates with the broader goal of self-efficacy identified in Massaty et al.'s review [28] but achieves it via a socio-cognitive mechanism rather than purely technical assistance.

5.4 Limitations and Future Work

While this study offers promising evidence for the benefits of personality-aware tools, several contextual aspects are worth noting. This pilot implementation involved 30 participants, providing a focused foundation that future research can extend across larger and more diverse student populations. By evaluating Codenote as an integrated instructional environment, this work establishes a baseline for further exploring the synergistic interactions between its various intelligent components. Additionally, the system's innovative use of non-invasive behavioral markers for personality inference opens the door for future studies to incorporate traditional assessments as a means of providing multi-dimensional validation. The stable role assignment used in this intervention helped ensure experimental consistency, though future iterations involving role rotation could provide broader insights into diverse self-efficacy gains. These considerations serve as valuable starting points for the ongoing optimization of AI-supported collaborative learning.

A limitation of the current experimental design is the absence of a 'randomly paired' control group. Due to the constrained class size (N = 30), dividing participants into three conditions would have compromised statistical power. Consequently, the study compared personality-based pairing against individual work. While this design leaves open the possibility that the observed benefits stem from collaboration itself rather than the specific matching strategy, prior literature suggests that pair programming does not inherently guarantee superior outcomes. Studies such as McChesney [16] and Demir and Seferoglu [17] have noted that without strategic grouping, pair programming can lead to frustration or unequal participation. Furthermore, the specific gain in adaptive self-efficacy in the absence of significant changes in persuasive or innovative dimensions suggests that the effect is not a generic 'collaboration bonus'. Instead, it points to the specific influence of working with a complementary partner, which requires students to actively adapt to differing perspectives, thereby isolating the mechanism of personality complementarity to a plausible degree. To move beyond this plausible isolation and establish rigorous causality, we explicitly commit to deploying a multi-arm experimental design in future large-scale studies, directly comparing AI-driven heterogeneous

pairing against a randomly paired control group. Furthermore, the current evaluation relies primarily on self-reported measures of self-efficacy to validate the psychological impact of the intervention. While fostering these affective states is a critical precursor to learning, we recognize the necessity of objective technical validation. In our future research trajectory, we explicitly commit to integrating objective performance indicators such as code completion rates, algorithmic efficiency, and compilation success frequencies. Combining these objective metrics with subjective psychological assessments will provide a comprehensive evaluation of the Codenote system's true pedagogical efficacy.

6 Conclusions

The present study validated the effectiveness of the Codenote system, specifically its AI-powered pair programming functionality, on students' self-efficacy across adaptive, innovative, and persuasive dimensions. The findings indicate that Codenote's automated pairing mechanism, which groups students with complementary personality traits, significantly enhances adaptive self-efficacy, suggesting that exposure to diverse perspectives and problem-solving approaches strengthens learners' confidence in managing complex and dynamic programming tasks. No significant improvements were observed in innovative or persuasive self-efficacy, implying that these dimensions may require additional system-embedded scaffolds such as opportunities for creative exploration, leadership roles, or structured reflection activities. Overall, the results highlight that the effectiveness of AI-supported pair programming is dimension-specific and contingent upon the quality of interaction and intelligent pairing strategies rather than collaboration alone. These findings contribute to the literature by clarifying the conditions under which AI-driven tools can enhance students' psychological readiness and adaptive capabilities. Future research should explore long-term effects, the integration of additional automated scaffolds, and the interaction patterns within pairs to further understand the mechanisms underlying the development of self-efficacy in AI-mediated environments.

7 Future Work

Based on the findings and limitations of the current study, several strategic directions for future research and system development have been identified.

1. Enhanced Instructional Scaffolding for Creative and Leadership Domains: While the current personality-based pairing successfully improved adaptive self-efficacy, the lack of significant gains in innovative and persuasive dimensions suggests the need for more explicit pedagogical interventions. Future iterations of Codenote will integrate dynamic role rotation mechanisms. Rather than static assignments, the system could algorithmically prompt role swaps (Driver/Navigator) at critical project milestones. Furthermore, to bolster persuasive self-efficacy, the system could introduce structured "conflict resolution" scenarios where pairs must negotiate architectural decisions, supported by automated prompts that encourage the quieter partner to lead the discussion.
2. Integration of Generative AI for Intelligent Feedback: Building upon the Runtime Code Analysis module detailed in Section 3.3, future work aims to transition from rule-based validation to Generative AI-driven mentorship. By embedding Large Language Models (LLMs) into the AI Inspector, the system could move beyond error detection to provide conversational, Socratic-style guidance. This would allow Codenote not only to identify what is wrong but to explain why, customizing the complexity of the explanation based on the learner's detected proficiency level, thereby providing a more personalized scaffolding experience.
3. Multimodal Analysis of Collaboration Dynamics: To deepen our understanding of how complementary personality traits influence collaboration, future research should leverage the data collected by the Pair Programming Plugin. Specifically, applying Natural Language Processing (NLP) techniques to analyze

the chat logs between partners could reveal patterns in communication styles—such as negotiation frequency, sentiment, and turn-taking behavior. Correlating these linguistic markers with personality profiles would provide granular insights into the mechanisms underlying effective peer modeling, moving beyond outcome-based metrics to process-oriented analysis.

4. Longitudinal Scalability and Transferability: Finally, longitudinal studies are required to determine the durability of the observed self-efficacy gains. It remains to be seen whether the confidence built through personality-aligned pairing transfers to individual coding tasks or more complex software engineering challenges, such as backend development or system architecture design. Extending the experimental scope to include diverse student populations across different institutions will also be crucial in establishing the generalizability of the behavior-based pairing algorithm.

Acknowledgement: Not applicable.

Funding Statement: This research was partially supported by the National Science and Technology Council (NSTC), Taiwan, under Grant Nos.: 113-2410-H-155-023 and 114-2410-H-155-012-MY2. The grants were received by author Chun-Hsiung Tseng. The sponsor's website is available at https://www.nstc.gov.tw/.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Chun-Hsiung Tseng, Hao-Chiang Koong Lin, Andrew Chih-Wei Huang; system implementation: Jia-Rou Lin, Chun-Hsiung Tseng; data collection and experiment execution: Jia-Rou Lin; analysis and interpretation of results: Chun-Hsiung Tseng, Hao-Chiang Koong Lin, Andrew Chih-Wei Huang; draft manuscript preparation: Jia-Rou Lin, Chun-Hsiung Tseng. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author, Chun-Hsiung Tseng, upon reasonable request.

Ethics Approval: The study protocol was reviewed and approved by the Human Research Ethics Governance & Ethical Review Committee of National Cheng Kung University (Case No.: 112-193). Please note that as Yuan Ze University does not maintain an internal Institutional Review Board (IRB), the ethical review process was officially delegated to and approved by the aforementioned independent committee, which is a standard practice for human subject research in Taiwan. Prior to the experiment, all participants provided written informed consent. They were explicitly informed that their operational data and note-taking behaviors would be analyzed for research purposes only, and that their participation was voluntary. To ensure privacy and confidentiality, all personal identifiers were removed and replaced with pseudo-anonymized IDs during the data analysis process. The behavior-based personality inference was strictly used for grouping purposes within the instructional activity and did not influence students' course grades or academic evaluations.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Huang KH. Computer education reform in Taiwan. Nat Sci Educ A Compr Sch. 2024;30(1):14–23. doi:10.48127/gu/24.30.14.
2. Chen JM, Wu TT, Sandnes FE. Exploration of computational thinking based on bebras performance in webduino programming by high school students. In: Wu TT, Huang YM, Shadiev R, Lin L, Starčič AI, editors. Innovative technologies and learning. Cham, Switzerland: Springer International Publishing; 2018. p. 443–52. doi:10.1007/978-3-319-99737-7_47.
3. Dwan F, Oliveira E, Fernandes D. Predição de zona de aprendizagem de alunos de introdução à programação em ambientes de correção automática de Código. Anais Do XXVIII Simpósio Brasileiro De Informática Na Educ SBIE 2017. 2017;1:1507. doi:10.5753/cbie.sbie.2017.1507.

4. Williams L, Wiebe E, Yang K, Ferzli M, Miller C. In support of pair programming in the introductory computer science course. Comput Sci Educ. 2002;12(3):197–212. doi:10.1076/csed.12.3.197.8618.
5. VanDeGrift T. Coupling pair programming and writing: learning about students' perceptions and processes. In: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education; 2004 Mar 3–7; Norfol, VA, USA. New York, NY, USA: ACM. p. 2–6. doi:10.1145/971300.971306.
6. Chigona W, Pollock M. Pair programming for information systems students new to programming: students' experiences and teachers' challenges. In: Proceedings of the PICMET'08—2008 Portland International Conference on Management of Engineering & Technology; 2008 Jul 27–31; Cape Town, South Africa. New York, NY, USA: IEEE. p. 1587–94. doi:10.1109/PICMET.2008.4599777.
7. Poonam R, Yasser CM. An experimental study to investigate personality traits on pair programming efficiency in extreme programming. In: Proceedings of the 2018 5th International Conference on Industrial Engineering and Applications (ICIEA); 2018 Apr 26–28; Singapore. New York, NY, USA: IEEE; 2018. p. 95–9. doi:10.1109/IEA.2018.8387077.
8. Mairesse F, Walker MA, Mehl MR, Moore RK. Using linguistic cues for the automatic recognition of personality in conversation and text. J Artif Intell Res. 2007;30:457–500. doi:10.1613/jair.2349.
9. Meidenbauer KL, Niu T, Choe KW, Stier AJ, Berman MG. Mouse movements reflect personality traits and task attentiveness in online experiments. J Pers. 2023;91(2):413–25. doi:10.1111/jopy.12736.
10. Martínez Mejía RD, Rodríguez Villanueva BP. Key factors influencing initial learning in computer programming. Rev Científica Sist E Informática. 2024;4(2):3.
11. Stajkovic AD, Bandura A, Locke EA, Lee D, Sergent K. Test of three conceptual models of influence of the big five personality traits and self-efficacy on academic performance: a meta-analytic path-analysis. Pers Individ Differ. 2018;120:238–45. doi:10.1016/j.paid.2017.08.014.
12. Verbree AR, Maas L, Hornstra L, Wijngaards-de Meij L. Personality predicts academic achievement in higher education: differences by academic field of study? Learn Individ Differ. 2021;92:102081. doi:10.1016/j.lindif.2021.102081.
13. Brandt ND, Lechner CM, Tetzner J, Rammstedt B. Personality, cognitive ability, and academic performance: differential associations across school subjects and school tracks. J Pers. 2020;88(2):249–65. doi:10.1111/jopy.12482.
14. Ainsworth S, Th Loizou A. The effects of self-explaining when learning with text or diagrams. Cogn Sci. 2003;27(4):669–81. doi:10.1016/S0364-0213(03)00033-8.
15. McDowell C, Werner L, Bullock HE, Fernald J. Pair programming improves student retention, confidence, and program quality. Commun ACM. 2006;49(8):90–5. doi:10.1145/1145287.1145293.
16. McChesney I. Three years of student pair programming: action research insights and outcomes. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education; 2016 Mar 2–5; Memphis, TN, USA. New York, NY, USA: ACM; 2016. p. 84–9. doi:10.1145/2839509.2844565.
17. Demir Ö, Seferoglu SS. The effect of determining pair programming groups according to various individual difference variables on group compatibility, flow, and coding performance. J Educ Comput Res. 2021;59(1):41–70.
18. Kovari A, Katona J. Effect of software development course on programming self-efficacy. Educ Inf Technol. 2023;28(9):10937–63. doi:10.1007/s10639-023-11617-8.
19. Avcu YE, Ayverdi L. Examination of the computer programming self-efficacy's prediction towards the computational thinking skills of the gifted and talented students. Int J Educ Methodol. 2020;6(2):259–70. doi:10.12973/ijem.6.2.259.
20. Schultz O, Blaszczyk T. Introduction of process in embedded programming supporting students' self-efficacy—case study. In: Towards a new future in engineering education, new scenarios that european alliances of tech universities open up. Barcelona, Spain: Universitat Politècnica de Catalunya; 2022. p. 1610–8. doi:10.5821/conference-9788412322262.1200.
21. El Khoury J, Safa N, Khoury J, Nasrallah R. Measuring the relationship between self-efficacy beliefs and performance attainments of first year engineering students in the programming course. TechRxiv. 2023. doi:10.36227/techrxiv.24639594.v1.

22. Ng TWH, Lucianetti L. Within-individual increases in innovative behavior and creative, persuasion, and change self-efficacy over time: a social-cognitive theory perspective. J Appl Psychol. 2016;101(1):14–34. doi:10.1037/apl0000029.
23. Chao J, Atli G. Critical personality traits in successful pair programming. In: Proceedings of the AGILE 2006 (AGILE'06); 2006 Jul 23–28; Minneapolis, MN, USA. New York, NY, USA: IEEE; 2006. p. 5–93. doi:10.1109/AGILE.2006.20.
24. Hannay JE, Arisholm E, Engvik H, Sjoberg DIK. Effects of personality on pair programming. IEEE Trans Softw Eng. 2010;36(1):61–80. doi:10.1109/TSE.2009.41.
25. Karimi Z, Baraani-Dastjerdi A, Ghasem-Aghaee N, Wagner S. Influence of personality on programming styles an empirical study. J Inf Technol Res. 2015;8(4):38–56. doi:10.4018/jitr.2015100103.
26. Haenggli M, Hirschi A. Career adaptability and career success in the context of a broader career resources framework. J Vocat Behav. 2020;119(1):103414. doi:10.1016/j.jvb.2020.103414.
27. Yilmaz R, Karaoglan Yilmaz FG. The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. Comput Educ Artif Intell. 2023;4:100147. doi:10.1016/j.caeai.2023.100147.
28. Massaty MH, Fahrurozi SK, Budiyanto CW. The role of AI in fostering computational thinking and self-efficacy in educational settings: a systematic review. Indones J Inform Educ. 2024;8(1):49. doi:10.20961/ijie.v8i1.89596.