A Camera-Assisted C++ Learning Software with Recognition for Cognitive Strategy Addressing Student Self-Handicapping Behavior

by

Mendoza, Lance Rendell D.

Escalada, John Edwin V.

Florendo, Von Venette C.

Viloria, Ronniel John M.

Submitted in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Computer Science

at

FEU Institute of Technology

Month/Year

Thesis Adviser

APPROVAL AND ACCEPTANCE SHEET

The thesis entitled **“Thesis Title, Bold, Times New Roman 12”** prepared and submitted by:

Name of Students

Arrange Alphabetically

Line Spacing Multiple 1.15

In partial fulfillment of the course of requirement for the Degree of Bachelor of Science in(Program)has been examined and is hereby recommended for approval.

Name of Panel 1	Name of Panel 2
Panelist 1	Panelist 2

Name of Head Panel

Head Panelist

Accepted as partial fulfillment of the requirements for the **Degree of Bachelor of Science in Computer Science.**

Dr. Shaneth C. Ambat/ Ms. Eliss Malasaga	Name of Mentor
Thesis Advisers	Thesis Mentor

Dr. Shaneth C. Ambat

Program Director

November ______, 2022

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

UBELT	University Belt

Chapter 1

INTRODUCTION

In programming education, many students exhibit self-handicapping behaviors such as procrastination, reduced effort, and cognitive disengagement, often triggered by fear of failure, low self-efficacy, or difficulty managing cognitive load. These behaviors serve as self-protective mechanisms to preserve one's sense of competence but ultimately hinder learning persistence, problem-solving ability, and long-term skill development. In challenging subjects like C++, where abstract reasoning and logical structuring are required, such tendencies become even more prevalent and detrimental to academic progress. Addressing these behavioral and emotional barriers is crucial to fostering effective learning and improving programming performance among students.

To address this issue, the present study proposes the development of a Camera-Assisted C++ Learning Software with Cognitive Strategy Recognition designed to detect and respond to self-handicapping behaviors in real time. The system integrates video-based emotion analysis with student interaction data, such as task completion time, error frequency, and coding behavior, using machine learning and ensemble modeling. By interpreting these data streams, the system can deliver adaptive, personalized feedback that encourages engagement, persistence, and productive learning strategies. Through this approach, the study aims to create an intelligent and emotion-aware learning environment that supports both the cognitive and affective aspects of programming education.

1.1. Background of the Study

Programming, especially C++, is widely recognized as one of the most cognitively demanding subjects in computer science(Peitek et al., 2023). Unlike purely theoretical courses, learning to code requires sustained problem-solving, logical reasoning, and the ability to apply abstract concepts in practical situations(Abran et al. 2020). This complexity often leads students to experience cognitive strain, frustration, or self-doubt. When faced with these challenges, many learners unconsciously adopt self-handicapping behaviors, such as procrastination, making excuses, or reducing effort, to protect their self-esteem from potential failure (Mendoza et al., 2020). These maladaptive coping strategies interfere with learning persistence and performance, making them an important focus for intelligent tutoring systems (ITS) that aim to foster effective learning behavior(Mutawa & Sruthi, 2023; Salloum et al., 2025).

Traditional programming tutorials and online learning platforms primarily focus on cognitive instruction, such as syntax, problem sets, and code compilation. However, they often overlook the affective and behavioral dimensions of learning. Recent studies emphasize that addressing emotional states, such as frustration, boredom, or anxiety, can improve student engagement and cognitive processing (Mutawa & Sruthi, 2023; Salloum et al., 2025). As a result, there is growing interest in emotion-aware and behavior-adaptive learning systems, particularly within the domain of artificial intelligence in education (Alam, 2023; Lin, Huang, & Lu, 2023).

Self-handicapping behaviors manifest in both cognitive strategies (e.g., negative self-talk, self-doubt) and overt physical actions (e.g., sighing, slouching, disengaging from the screen). According to Yıldırım and Demir (2019), procrastination, test anxiety, and avoidance are among the strongest predictors of self-handicapping, while emotional regulation and self-esteem act as protective factors. Locally, Erezo et al. (2020) observed that Filipino students frequently engage in excuse-based avoidance (e.g., citing internet problems or fatigue) as a means to maintain self-worth in challenging learning contexts. These findings suggest that overt and observable signs of disengagement can serve as behavioral indicators of self-handicapping, signs that can be captured and analyzed by modern computer-vision systems.

The integration of camera-based emotion recognition provides a powerful means of detecting these behaviors in real time. Liu, Jiang, and Jiang (2025) explain that live video input enables systems to capture subtle cues such as facial expressions, eye gaze, and posture, which are essential for recognizing cognitive fatigue, avoidance, or frustration(Salloum et al., 2025). Without such visual data, an ITS cannot fully infer the learner's affective or attentional state. Combining this capability with AI-based analysis allows the system to respond adaptively, offering motivational feedback, adjusting difficulty, or suggesting rest, to minimize self-handicapping tendencies(Unciti et al., 2024). Within the local context, Bringula (2019) and Estrellado & Miranda (2023) highlight the lack of culturally adaptive ITS systems that incorporate affective computing in the Philippines. They advocate for emotion-aware platforms that can deliver personalized, ethical, and responsive feedback. Integrating such systems not only enhances learner motivation but also supports national goals for digital transformation in education.

From a cognitive perspective, problem-solving in C++ continues to be a relevant testbed for this kind of system. Naveed (2021) demonstrated that C++'s performance advantages and its low-level data manipulation expose students to deep algorithmic thinking, offering an ideal environment to observe cognitive load, frustration, and persistence. Moreover, research by Zhong and Zhan (2025) and Kouam (2024) confirms that ITS frameworks improve novice learners' engagement and understanding through adaptive scaffolding and real-time feedback, making the programming domain particularly suitable for intelligent, emotion-sensitive tutoring interventions.

This study proposes the development of a camera-assisted C++ learning software equipped with cognitive strategy recognition. The system is designed to detect possible self-handicapping behaviors and deliver adaptive feedback to help students stay engaged. By combining traditional programming instruction with emotion-aware support, the project aims to provide a more balanced approach that addresses both the cognitive and emotional challenges of learning C++.

Building on these developments, the present study proposes the design and implementation of a Camera-Assisted C++ Learning Software with Cognitive Strategy Recognition for addressing student self-handicapping behavior. This system will integrate webcam-based emotion detection and interaction-based behavioral analysis through an ensemble of machine learning models to identify early signs of avoidance. Tailored, real-time adaptive feedback will then be provided to help learners re-engage, build resilience, and develop effective programming strategies. In doing so, the project aims to bridge the gap between technical instruction and emotional support, fostering deeper engagement and improved outcomes in C++ education.

1.2.Theoretical Framework of the Study

Intelligent Tutoring System

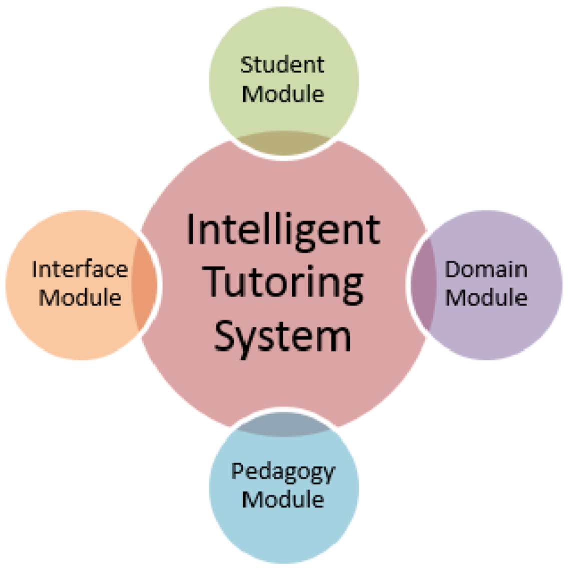


Figure 1.**Intelligent Tutoring System**
Reprinted from
Raza, A. (2020). INTELLIGENT TUTORING SYSTEMS AND METACOGNITIVE LEARNING STRATEGIES: a SURVEY. ResearchGate.
https://www.researchgate.net/publication/340842792_INTELLIGENT_TUTORING_SYSTEMS_AND_METACOGNITIVE_LEARNING_STRATEGIES_A_SURVEY

Intelligent Tutoring Systems (ITSs) use artificial intelligence to create adaptive virtual tutors that personalize learning experiences for each student by assessing their performance and needs. These systems are built on four core components; the Domain Model (organizing subject knowledge), the Student Model (tracking learner progress), the Tutoring Model (designing instructional actions), and the User interface Model (facilitating interaction). Together, these modules allow ITSs to deliver targeted feedback and dynamic instructional strategies that support effective learning. By integrating these elements, ITSs aim to emulate the benefits of one-on-one tutoring in an automated, scalable way.

Computer Vision

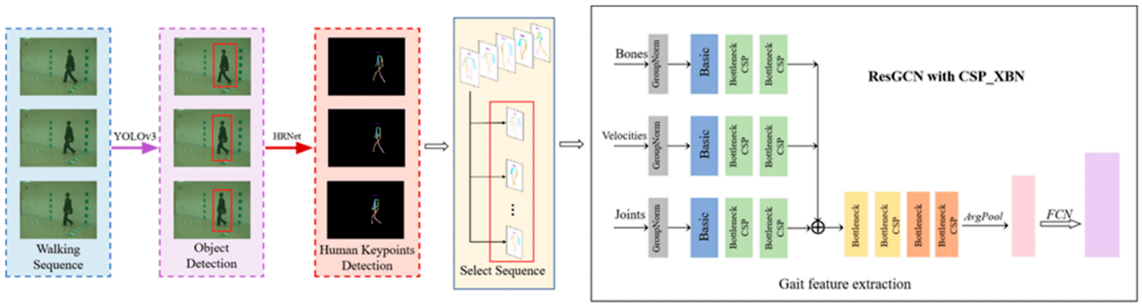


Figure 2. **Algorithm block diagram**
Reprinted from
Han, K., & Li, X. (2023). Research Method of Discontinuous-Gait Image Recognition Based on Human Skeleton Keypoint Extraction. Sensors, 23(16). *https://doi.org/10.3390/s23167274*

Computer vision is one of the many fields that make up computer technology, it focuses on giving computers the ability to analyze and comprehend visual data. Computer vision is utilized in your research to process and analyze video data in order to identify significant patterns, especially those pertaining to human movements. Specifically, the system utilizes computer vision for:

Deep Learning

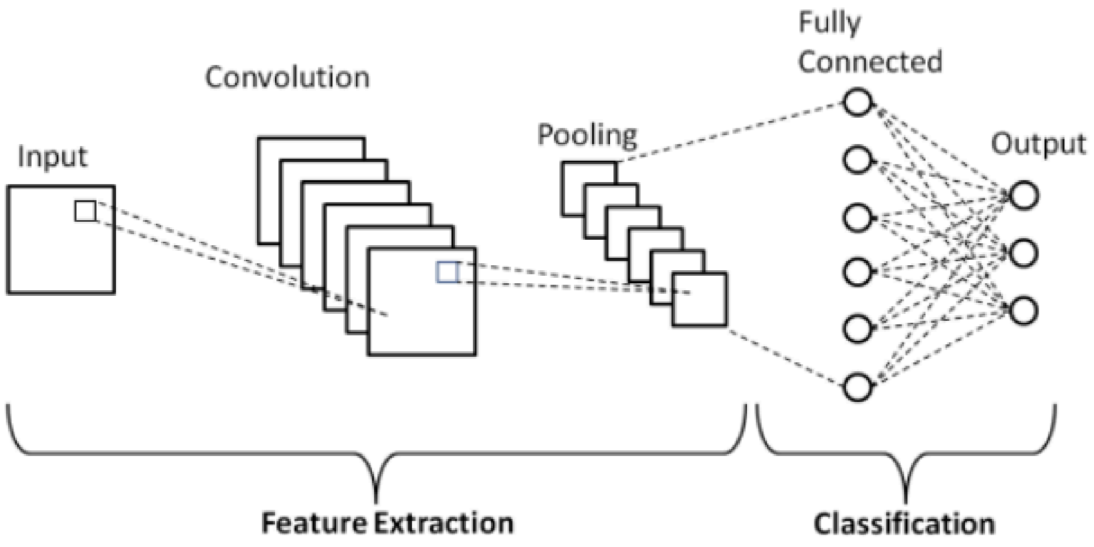


Figure ?. **Schematic diagram of a basic convolutional neural network (CNN) architecture.**

Reprinted from

Phung, V. H., & Rhee, E. J. (2019). A High-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. Applied Sciences (Switzerland), 9(21). https://doi.org/10.3390/app9214500

Deep learning is a branch of machine learning that uses neural network training algorithms to autonomously identify patterns and representations in large amounts of data. In contrast to conventional machine learning, it makes use of multi-layered algorithmic structures known as deep neural networks, which are capable of modeling extremely abstract data aspects. These networks consist of layers of interconnected nodes, or neurons, which process and transform input data to extract meaningful patterns without requiring manual feature engineering.

1.3.Conceptual Framework of the Study

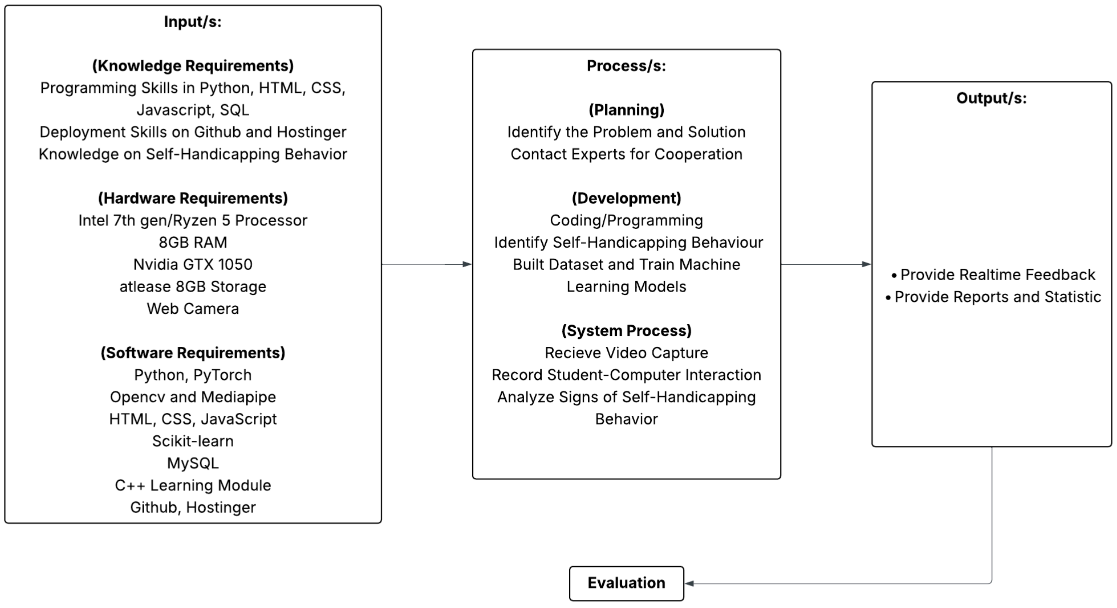

Input/s:
(Knowledge Requirements)
Programming Skills in Python, HTML, CSS, Javascript, SQL
Deployment Skills on Github and Hostinger
Knowledge on Self-Handicapping Behavior
(Hardware Requirements)
Intel 7th gen/Ryzen 5 Processor
8GB RAM
Nvidia GTX 1050
atlease 8GB Storage
Web Camera
(Software Requirements)
Python, PyTorch
Opencv and Mediapipe
HTML, CSS, JavaScript
Scikit-learn
MySQL
C++ Learning Module
Github, Hostinger
Process/s:
(Planning)
Identify the Problem and Solution
Contact Experts for Cooperation
(Development)
Coding/Programming
Identify Self-Handicapping Behaviour
Built Dataset and Train Machine Learning Models
(System Process)
Recieve Video Capture
Record Student-Computer Interaction
Analyze Signs of Self-Handicapping Behavior
Output/s:
• Provide Realtime Feedback
• Provide Reports and Statistic
Evaluation

Figure ?. **Conceptual Framework**

The conceptual framework serves as a foundation for developing and understanding complex systems and algorithms. It defines the key concepts, variables, and their interrelationships relevant to the study. In computer science, conceptual frameworks are often based on existing theories, models, and empirical findings that guide system design and implementation. For this study, the framework provides a structured approach for analyzing, designing, and developing the proposed system, ensuring that each component aligns with the objectives and effectively addresses the identified problem.

Input Phase

The input phase defines the essential knowledge, hardware, and software components necessary for the development and operation of the Cognitive Strategy Recognition System for Self-Handicapping Behavior Detection.

The knowledge requirements include proficiency in programming languages such as Python, HTML, CSS, JavaScript, and SQL, which are necessary for developing the web-based interface, backend processing, and machine learning models. Competence in GitHub and Hostinger deployment ensures version control and seamless web hosting. Additionally, understanding self-handicapping behavior is vital for defining behavioral indicators and labeling datasets used during model training.

The hardware requirements ensure smooth system execution and data processing. The system requires at least an Intel 7th Gen or Ryzen 5 Processor, 8GB of RAM, and an NVIDIA GTX 1050 GPU to handle machine learning computations and video analysis efficiently. A web camera is essential for capturing facial and behavioral data, while at least 8GB of storage is needed for maintaining datasets, logs, and trained model files.

The software requirements consist of Python and PyTorch for model development, OpenCV and MediaPipe for video processing and facial analysis, Scikit-learn for implementing the Random Forest and Logistic Regression models, and MySQL for managing interaction data. HTML, CSS, and JavaScript are used for creating the learning environment interface, while GitHub and Hostinger support deployment. The C++ Learning Module serves as the interactive platform where student activities and behaviors are monitored.

Process Phase

The process phase describes the step-by-step development workflow used to design, train, and implement the system.

In the planning phase, the research team identifies the core problem, detecting self-handicapping behaviors and establishes a framework for integrating both video-based and interaction-based analysis. Collaboration with experts in psychology and education ensures that behavioral indicators align with validated definitions of self-handicapping.

During the development phase, system coding and integration are carried out. A dataset is constructed from student interactions and video inputs, which is then used to train the Random Forest model for behavioral recognition. Simultaneously, the video recognition component is developed to analyze facial and physical cues potentially associated with cognitive disengagement.

The system process involves capturing live video input and logging user interactions in real time. Both data streams are analyzed , the video model identifies visual signs of self-handicapping, while the behavioral model detects interaction-based patterns such as prolonged inactivity, frequent resets, or minimal code changes. The outputs from these models are then combined through an ensemble using logistic regression, which enhances detection accuracy by synthesizing both perspectives into a single decision.

Output Phase

The output phase focuses on the system's deliverables and generated insights. The system provides real-time feedback to students based on detected behavioral or cognitive disengagement patterns. This feedback may include motivational prompts, task-specific hints, or encouragement messages aimed at helping the learner re-engage with the material and overcome self-handicapping tendencies.

Additionally, the system compiles reports and statistical summaries, including behavioral trends, engagement levels, and system accuracy metrics. These insights allow educators and researchers to assess both student progress and the effectiveness of the model.

Evaluation phase

Finally, the evaluation phase focuses on assessing both the performance and software quality of the system. Quantitative evaluation will utilize standard machine learning metrics such as Accuracy, Precision, Recall, F1-Score, Confusion Matrix, and AUC/ROC to measure the effectiveness and reliability of the model in detecting self-handicapping behaviors. These metrics ensure that the recognition system performs consistently under various user interactions and video input conditions.

Beyond model accuracy, the system's overall quality will be assessed following the ISO/IEC 25010 categorization of software quality requirements (Rebeś & Rebeś, n.d.). This framework evaluates eight key characteristics: Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability. Each criterion ensures that the system is not only technically sound but also practical, scalable, and user-centered for deployment in educational environments.

This dual-layer evaluation, combining machine learning performance with software quality assessment, ensures that the developed Cognitive Strategy Recognition System meets both analytical accuracy and international software quality standards, making it a dependable tool for real-world Intelligent Tutoring Systems.

1.4. Statement of the Problem

Programming is one of the most challenging subjects for computer science students, requiring persistence and strong problem-solving skills. However, many learners adopt self-handicapping behaviors such as procrastination, reduced effort, or avoidance, often triggered by anxiety, fear of failure, and low self-confidence (Mendoza et al., 2020). These behaviors limit engagement and hinder the development of programming competence. The study intends to answer the following questions:

1. What parameters will be used to determine the self handicapping behaviors?
2. How will the system classify self-handicapping behavior?
3. How will the ITS provide tailored feedback for self-handicapping behavior?
4. How will the system be evaluated based on the system's performance, effectiveness and acceptance?

1.5. Objectives of the Study

The main objective of this study is to develop an Intelligent Tutoring System (ITS) capable of detecting Self-Handicapping Behavior within a web-based C++ learning environment. The system aims to utilize machine learning-based behavioral recognition and video analysis to identify overt signs of self-handicapping among learners during system interaction. Furthermore, it seeks to enhance the understanding of students' cognitive and emotional engagement, thereby fostering a learning environment that promotes motivation, self-awareness, and meaningful academic support.

1. Identify and define the key behavioral and emotional parameters that indicate self-handicapping tendencies among students during C++ learning activities.
2. Design a constraint-based algorithm model and an ensemble framework to classify self-handicapping behavior among students, guided by expert recommendations and behavioral analysis.
3. Deliver adaptive feedback, as advised by professionals, during C++ learning sessions to improve engagement.
4. Evaluate system performance using ROC curves, Confusion Matrix, and usability metrics from the Technology Acceptance Model (TAM) and ISO 20510 standards.

1.6. Scope and Limitations of the Study

Scope

This study focuses on developing a Cognitive Strategy Recognition System designed to detect self-handicapping behaviors among students in a C++-based Intelligent Tutoring System (ITS) environment.

Video data - captured through a webcam, to identify observable cognitive indicators such as confusion, disengagement, or avoidance; and

User interaction data - task completion time, frequency of edits, response intervals, and code submission patterns.(logs)

A Random Forest model analyzes behavioral interaction data, while a video-based recognition model processes facial and physical cues. These models are combined using an ensemble approach via logistic regression to improve detection accuracy. The output provides feedback and performance analytics to help address self-handicapping tendencies and promote more effective learning strategies within the C++ learning environment.

Delimitations

The system is designed specifically as a web application for C++ programming exercises and focuses exclusively on selected C++ topics. It may not generalize to other programming languages or subjects. Data collection is limited to video and interaction logs within the ITS; no biometric sensors or affective data (e.g., heart rate, galvanic response) are used. The study focuses on cognitive strategy and self-handicapping detection, not on comprehensive emotional or motivational analysis. Feedback provided by the system is limited to adaptive hints, encouragement messages, and behavioral insights, rather than personalized tutoring or content generation. Testing and evaluation are conducted in a controlled laboratory or academic setting, not in open online environments.

Limitations

The accuracy of video-based recognition may vary due to lighting conditions, camera quality, or user positioning during learning sessions. Real-time feedback may introduce minor performance delays depending on processing speed and available hardware resources. The system's interpretability depends on expert-defined rules and labeled datasets, which could introduce subjectivity during model training and evaluation.

1.7. Significance of the Study

This addresses both the cognitive and emotional challenges faced by students in learning C++. By integrating emotion recognition into a learning system, the project introduces a more personalized and supportive approach to programming education. The findings and output of this study will benefit several groups:

Students. The proposed system can help learners manage frustration and disengagement by providing adaptive feedback during programming sessions. This may encourage persistence, improve motivation, and ultimately enhance their understanding of C++.

Educators. Teachers may gain insights into students' emotional states while they work on programming tasks. This can guide them in adjusting their teaching strategies and offering targeted support for students who struggle with self-handicapping behaviors.

Academic Institutions. Schools can use the system as an innovative supplement to traditional teaching methods. By addressing both technical skills and emotional needs, institutions can improve student performance and keep them more engaged in more programming classes.

Psychologist. Psychologists play a critical role in educational settings by assessing students' learning and emotional needs and by developing interventions to support their educational, social, and psychological development. Their expertise helps teachers understand students better, allowing for tailored teaching strategies and creating supportive learning environments that address individual challenges, such as self-handicapping behaviors, to enhance students' success.

Future Researchers. This study can serve as a reference for further development of emotion-aware educational tools, not only for C++ but also for other programming languages and technical subjects where student motivation and persistence are critical.

1.8.Definition of Terms

1.8.1. Technical

Self-handicapping - a psychological strategy where individuals create or claim obstacles to protect self-esteem in anticipation of failure.

Overt self-handicapping - observable behaviors such as procrastination, reduced effort, or avoidance that impair performance.

Self-efficacy - the belief in one’s ability to succeed in specific tasks or domains.

Test anxiety - emotional distress experienced before or during assessments, often linked to performance avoidance.

Self-regulatory learning strategies - techniques used by learners to plan, monitor, and evaluate their learning processes (e.g., goal-setting, time management).

C++ - a high-performance, multi-paradigm programming language used for systems, applications, and educational software.

OpenCV - an open-source computer vision library widely used for image and video processing in C++ environments.

Standard Template Library (STL) - a collection of C++ template classes for data structures and algorithms, supporting generic programming.

Pointer - a variable that stores the memory address of another variable, central to low-level programming in C++.

Facial Emotion Recognition - the process of identifying emotional states from facial expressions using computer vision algorithms.

FER2013 Dataset - a benchmark dataset of labeled facial expressions used to train emotion recognition models.

Convolutional Neural Network (CNN) - a deep learning architecture optimized for image classification and pattern recognition tasks.

Frame Rate - the number of video frames processed per second, critical for real-time emotion and behavior detection.

Intelligent Tutoring System (ITS) - a computer-based learning system that adapts instruction based on a learner’s performance and behavior.

Adaptive Feedback - instructional responses tailored to a learner’s cognitive and emotional state.

Scaffolding - structured support provided to learners to help them master complex tasks gradually.

Interaction Logs - data collected from user actions (e.g., clicks, time-on-task) used to infer engagement and behavioral patterns.

1.8.2. Operational

Chapter 2

REVIEW OF RELATED LITERATURE AND STUDIES

This chapter will discuss the related literature and studies done by the researchers and the possible algorithms that can be used for this proposed system.

1.1. Local Related Literature

1.1.1

Mendoza et al. (2020) explored the emotional and behavioral patterns of students who engage in self-handicapping. Through focus group discussions, they found that students often procrastinate, reduce effort, or avoid tasks as a way to protect their self-worth in the face of academic challenges. Emotional triggers such as anxiety, fear of failure, and low self-confidence were central to these behaviors. Their findings underscore the importance of educational interventions that address both cognitive and emotional dimensions of learning an approach that aligns with the goals of emotion-aware tutoring systems.

1.1.2

Bringula (2019) reviewed the state of ITS development in the Philippines. He identified key limitations such as the lack of culturally responsive systems, minimal interdisciplinary collaboration, and limited integration of affective computing. Bringula emphasized the need for ITS designs that reflect Filipino learning styles and incorporate emotional feedback mechanisms. His work supports the rationale for developing localized, emotion-aware tutoring systems that can adapt to the unique needs of Filipino learners.

1.1.3

Estrellado and Miranda (2023) examined the ethical, infrastructural, and pedagogical implications of AI integration in Philippine education. They highlighted the potential of AI to deliver personalized learning and adaptive feedback, while cautioning against challenges such as the digital divide, data privacy concerns, and limited faculty readiness. Their recommendations include strengthening policy frameworks and promoting interdisciplinary research to ensure ethical and effective AI deployment. This literature reinforces the importance of designing emotion-aware educational tools that are both technically sound and socially responsible.

1.2. Foreign Related Literature

1.2.1

Yıldırım and Demir (2019) conducted a correlational study which examined the psychological predictors of academic self-handicapping. Their findings revealed that procrastination and test anxiety were positively associated with self-handicapping behaviors, while self-esteem and self-compassion served as protective factors. The study emphasized that students with low emotional regulation and high avoidance tendencies are more likely to engage in self-handicapping to preserve self-worth. This research supports the emotional dimension of your thesis, particularly the need to detect and respond to affective states that may trigger disengagement in programming tasks.

1.2.2

Izadpanah and Charmi (2022) explored how social media usage influences self-handicapping behaviors. Using structural equation modeling, they found that excessive engagement with social networks negatively impacts academic achievement by fostering self-handicapping tendencies. However, self-regulatory learning strategies were shown to mediate this effect, suggesting that students with strong metacognitive skills can buffer against the negative consequences. This study reinforces the importance of integrating motivational and regulatory feedback mechanisms into intelligent tutoring systems to counteract behavioral avoidance.

1.2.3

Kouam (2024) conducted a mixed-methods evaluation of ITS adaptability across novice, intermediate, and advanced learners. The study found that ITS platforms significantly improved learning outcomes for beginners by offering personalized feedback and scaffolding. However, the impact was less pronounced for advanced learners, who required more complex problem-solving support. These findings validate the use of ITS in introductory C++ programming environments and highlight the importance of tailoring feedback based on learner proficiency an approach central to your proposed system.

1.2.4

Zhong and Zhan (2025) developed and evaluated an ITS in their study wherein their system utilized lexical and syntactic analysis to assess student code and provide real-time feedback aimed at improving computational thinking. The ITS was found to be particularly effective for low-level learners, enhancing motivation and reducing cognitive load. The study also emphasized the role of informative tutoring feedback (ITF) in promoting self-directed learning. This aligns closely with your thesis objective of delivering adaptive, emotionally responsive feedback during C++ instruction.

1.3. Local Related Studies

1.3.1

Erezo et al. (2020), a research team from the City College of Angeles, conducted a phenomenological study titled *Academic Self-Handicapping Strategies of Students in a Community College*. Their investigation focused on the lived experiences of college students who engage in self-handicapping behaviors across both face-to-face and distance learning modalities. Through focus group discussions, they uncovered that students often resort to excuses such as internet issues or fabricated emergencies to preserve their academic image. The study emphasized that while these strategies offer temporary relief, they ultimately compromise academic performance and student-teacher relationships. The authors recommend fostering open communication and self-accountability to mitigate the negative effects of self-handicapping.

In a collaborative effort published by IGI Global, Bringula (2019) critically examines the state of ITS development in the Philippines, highlighting the lack of culturally responsive systems and limited interdisciplinary collaboration. Bringula advocates for the integration of affective computing and emotion-aware feedback mechanisms to enhance learner engagement. His work provides a strong theoretical foundation for emotion-sensitive educational technologies tailored to Filipino learners.

1.3.2

Estrellado and Miranda (2023) explores the ethical, infrastructural, and pedagogical implications of AI integration in Philippine classrooms. While acknowledging AI's potential for personalized learning and adaptive feedback, they caution against challenges such as the digital divide and data privacy concerns. The authors call for robust policy frameworks and interdisciplinary research to ensure responsible AI deployment. Their insights reinforce the importance of designing emotion-aware systems that are both technically sound and socially grounded.

1.3.3

Go et al. (2020), a team of education researchers based in Cebu, examined the relationship between emotional intelligence and teaching performance. Surveying 160 public school teachers, they found that emotional regulation and the ability to separate personal issues from professional responsibilities significantly enhance instructional effectiveness. This study underscores the role of emotional awareness in educational success and supports the integration of emotion recognition technologies in learning environments.

1.3.4

Santamaría-Bonfil, Ibáñez, Pérez-Ramírez, Arroyo-Figueroa et al. (2020) conducted a study in which they used learning analytics (LA) to model student performance in a Virtual Reality Training System (VRTS) for lineworkers. Using trace data from 1,399 trainees across 329 training courses over eight years, they built machine learning classifiers (including Random Forest) to distinguish between "trained" and "untrained" participants based on logs of errors, profile, and course variables. They also performed feature importance analysis to identify which trace features (error types, step/substep performance, etc.) most strongly predicted final performance. A visualization tool allowed experts to spot error patterns, confusion, and misconceptions among trainees not observable purely via summary statistics. This is relevant to our study: it supports using interaction logs plus behavioral / error features to detect underlying learner difficulties, which parallels our goal of recognizing self-handicapping or cognitive strategy behavior in problem-solving environments, and suggests that trace-based ML + expert interpretation is a strong approach.

1.4. Foreign Related Studies

1.4.1

Abu-Naser (2008), a researcher from Al-Azhar University, developed a system, named CPP-Tutor, which was designed to assist learners in mastering C++ programming through adaptive feedback and structured instructional modules. The study emphasized the importance of ITS in improving student performance, particularly in subjects requiring procedural logic and syntax mastery. Abu-Naser's work provides a foundational model for integrating intelligent tutoring into programming education and supports the technical feasibility of your proposed system.

1.4.2

In a recent publication in *Smart Learning Environments*, Salloum et al. (2025) explored the integration of emotion recognition into educational technology.

Utilizing Convolutional Neural Networks (CNNs) trained on the FER2013 dataset, their system identified emotional states such as boredom, frustration, and confusion, and adapted instructional strategies accordingly. The results demonstrated improved learner engagement and retention, validating the role of affective computing in intelligent tutoring systems. This study directly supports the emotional feedback mechanism central to your thesis.

1.4.3

Yıldırım and Demir (2019) conducted a research which revealed that procrastination and test anxiety were positively associated with self-handicapping behaviors, while self-esteem and self-compassion served as protective factors. The study emphasized the emotional and cognitive dimensions of self-handicapping, reinforcing the need for educational interventions that address both psychological and behavioral factors an approach central to your proposed system.

1.4.4

Izadpanah and Charmi (2022), using structural equation modeling, demonstrated that excessive use of social media negatively impacts academic performance by fostering self-handicapping tendencies. However, students with strong self-regulatory learning strategies were able to mitigate these effects. This study underscores the importance of promoting metacognitive skills and adaptive feedback in educational systems, aligning with your goal of designing a tutoring platform that responds to both emotional and behavioral indicators of disengagement.

1.4.5

Liu, Jiang, and Jiang (2025) explain that embedding a camera in an ITS is crucial because only video can capture the subtle, real‑time cues, like facial expressions, eye gaze, and body posture, that signal whether a student is engaged or frustrated. Traditional ITS metrics (clicks, time stamps) miss these moment‑to‑moment behaviors. Cameras provide high‑frame‑rate (20–30 fps) video streams, enabling low‑latency emotion and attention classification so the system can adapt instruction immediately. In short, without a camera's live visual feed, an ITS cannot reliably detect and respond to learners' emotional and attentional shifts.

1.4.6

Naveed (2021) directly benchmarks C++ against Java on a suite of introductory algorithms, sorting, searching, and basic graph traversals, to explore which language offers the clearest pedagogical and performance advantages. The results show that C++ implementations consistently outpace their Java counterparts by 15–30 % in execution time and use up to 40 % less memory, thanks to C++'s lack of garbage‑collection pauses and its ability to allocate and free resources deterministically. Beyond raw speed, Naveed highlights how C++'s support for pointers and manual memory management exposes students to low‑level data structures and algorithm optimization strategies that remain hidden in Java's managed runtime. It is also pointed out that the Standard Template Library (STL) gives learners a powerful yet transparent toolkit for generic programming, in contrast with Java's type‑erased generics, which offer less compile‑time optimization and obscure the link between algorithm and data representation. Taken together, these findings argue that C++ not only trains students in core computer‑science concepts more rigorously but also ensures that their code runs with the efficiency modern ITS frameworks demand. (Naveed, 2021)

1.4.7

Abe, Fukawa, and Tanaka (2020) developed a visual programming environment designed to help novice programmers overcome difficulties, or "stumbling," in learning the C language. The system incorporates block-based editing and visual execution tracing, allowing learners to step through their code and observe variable and expression evaluations. Their findings showed improved task accuracy and understanding among students using the tool compared to traditional coding environments. This study highlights the role of interactive visualization and feedback in reducing cognitive barriers during programming, supporting the present research's aim to identify and address self-handicapping behaviors through behavioral and interaction-based recognition within problem-solving environments.

1.4.8

Karaci (2018) developed an Intelligent Tutoring System (ITS) utilizing a fuzzy logic decision framework combined with a constraint-based student model to teach punctuation in Turkish. The system evaluated student knowledge using certainty factors, number of attempts, and fuzzy logic reasoning to interpret uncertain learning states. Through adaptive feedback and individualized assessment, it provided personalized guidance and improved student engagement. The study highlights how integrating intelligent modeling and behavior-based analysis can effectively identify learning difficulties and support adaptive instruction. This aligns with the present study's objective of detecting self-handicapping behaviors and cognitive strategies through system interactions, demonstrating the value of data-driven ITS models in enhancing personalized learning experiences.

1.4.9

Khodeir, Wanas, and Elazhary (2022) developed an Intelligent Tutoring System that applies constraint-based student modeling to teach probability story problems, integrating scaffolding techniques to enhance learning adaptability. The system identifies learner misconceptions by analyzing constraint violations, specific logical or procedural errors students make while solving problems, and provides corrective feedback tailored to the detected issue. As the learner progresses, the system gradually reduces scaffolding support, promoting self-reliance and deeper conceptual understanding. The authors emphasize that such adaptive systems can effectively personalize instruction by continuously monitoring learner behavior and dynamically adjusting feedback intensity. Their research demonstrates how student modeling grounded in cognitive theory can detect strategic learning difficulties, such as avoidance or over-reliance on hints, paralleling this study's focus on recognizing cognitive strategies and self-handicapping behaviors within problem-solving environments.

1.4.10

The study Prescriptive Analytics: Literature Review and Research Challenges explores the transition of analytics paradigms, from descriptive, which summarizes what happened, and predictive, which forecasts what might happen, to prescriptive analytics, which determines the optimal actions to take based on data, constraints, and objectives. The authors emphasize that prescriptive analytics combines optimization, simulation, and machine learning techniques to generate data-driven recommendations rather than mere insights. This approach helps decision-making systems autonomously identify the best course of action given complex and dynamic scenarios. In the context of Intelligent Tutoring Systems (ITS), this concept provides a strong foundation for adaptive and personalized feedback mechanisms, where the system not only detects learner behaviors, such as hesitation, self-handicapping, or repeated misconceptions, but also determines the most effective response, whether through hints, scaffolding, or motivational reinforcement. By applying prescriptive analytics principles, the ITS can move beyond reactive assistance toward proactive pedagogical decision-making, enabling it to dynamically prescribe corrective or supportive strategies tailored to each student's cognitive and affective state.

1.4.11

Duran, Zavgorodniaia, and Sorva (2022) present a comprehensive review of Cognitive Load Theory (CLT) as applied to computing education, synthesizing empirical and theoretical work across programming, algorithmics, and software engineering instruction. They categorize cognitive load into intrinsic, extraneous, and germane components, and discuss how poorly designed materials (e.g., verbose code, redundant explanations) increase extraneous load, reducing students' capacity to engage in deep schema construction. The authors highlight evidence that effective instructional design (e.g., worked examples, code chunking, progressive complexity) can mitigate overload and promote learning efficiency. This review reinforces the foundation for your work: programming tasks inherently carry high intrinsic load, thereby amplifying the negative effects of self-handicapping strategies when students lack scaffolding. It supports your system's rationale in detecting behavioral signs of overload and intervening with just-in-time feedback or scaffolding to preserve learner cognitive capacity.

Chapter 3

METHODOLOGY

Methodology refers to the systematic framework and procedures used to design, develop, and evaluate the study, ensuring that its objectives are achieved with accuracy and consistency. It encompasses the processes of data collection, model training, analysis, and validation to maintain the reliability and replicability of results.

This chapter presents the methodological approach used in the development of the Cognitive Strategy Recognition Model for Self-Handicapping Behavior Detection in C++-Based ITS Learning Environments. The study focuses on identifying cognitive strategies linked to self-handicapping behaviors by integrating video-based emotion recognition and student interaction analysis within a C++ learning environment. It utilizes computer vision techniques to process visual input and Random Forest classification to interpret behavioral interaction patterns. These models are integrated through an ensemble learning approach using logistic regression as a meta-classifier to improve the accuracy and reliability of detection. The methodology also discusses the system architecture, data processing workflow, and evaluation metrics applied to assess the performance and effectiveness of the proposed model.

3.1.Type of Research

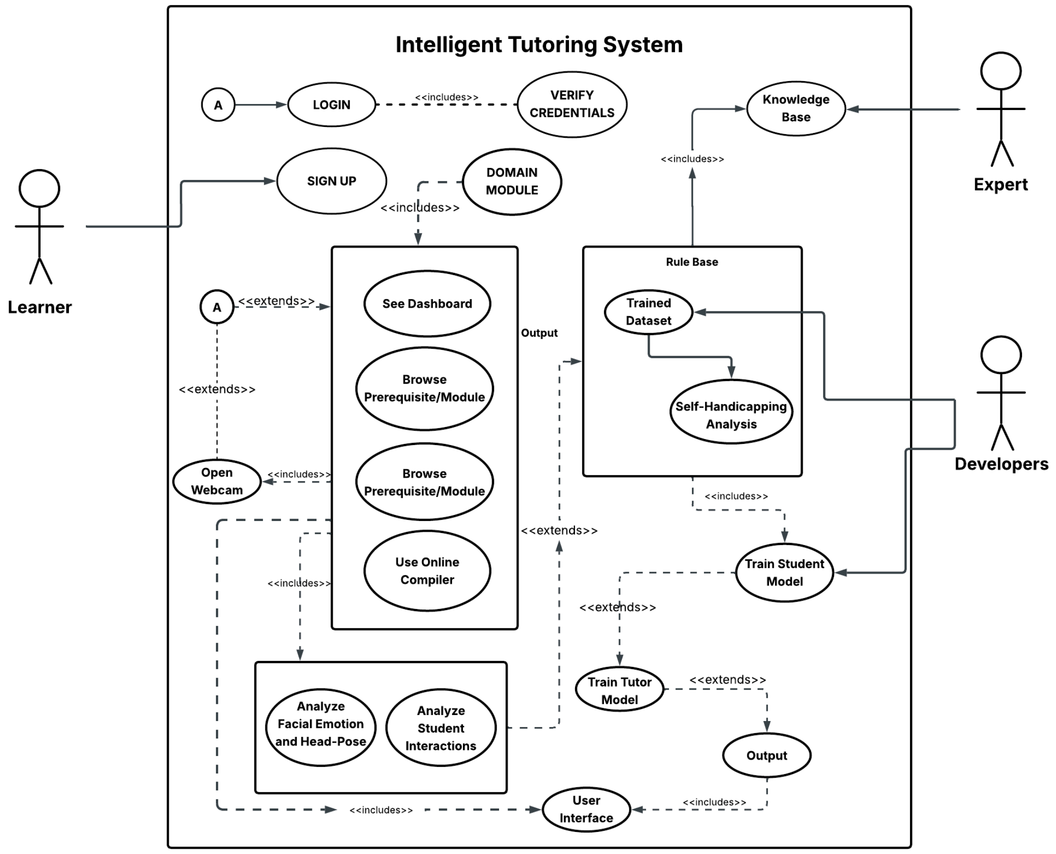
The prescriptive aspect of this research focuses on providing actionable solutions for improving learning outcomes by identifying and addressing self-handicapping behaviors in students. Prescriptive research aims to recommend or implement specific interventions to enhance performance and adapt student behavior toward optimal learning (Jonassen, 1997). By analyzing patterns of self-handicapping detected through system interaction, the study prescribes targeted feedback and guidance designed to foster persistence, reduce avoidance tendencies, and strengthen learning motivation (Alam, 2023). Such an approach ensures that the study contributes not only to understanding the behaviors but also to shaping systematic strategies for improving programming instruction effectiveness.

The developmental component of the study employs the Scrum methodology to design and iteratively improve a camera-assisted C++ learning system. Developmental research emphasizes the design, testing, and refinement of educational tools through structured, adaptive processes (Richey & Klein, 2014). In this study, Scrum practices enable incremental prototype development, where user feedback and observational data inform each system iteration (Schwaber & Sutherland, 2020). This cyclical approach aligns with developmental research principles, as it supports continuous enhancement of system functionality to better detect and respond to learners' behavioral and emotional states, thereby ensuring practical relevance and usability in authentic learning environments.

3.2.Project Design

To provide a clear understanding of the project's structure and implementation, this section presents four essential design components: the Activity Diagram, Use Case Diagram, Block Diagram, and System Architecture. These visual representations help illustrate how the Cognitive Strategy Recognition Model operates within an Intelligent Tutoring System (ITS) environment, supporting the detection of self-handicapping behaviors among students learning C++. The Activity Diagram outlines the system's workflow, showing the sequence of processes involved in recognizing and analyzing cognitive strategies. It visualizes how input data from student interactions are processed through the model to generate behavioral insights. The Use Case Diagram highlights the interactions between users such as students, instructors, and the system itself and the various system functions. It defines user roles and clarifies how each interacts with components like the recognition module, learning interface, and feedback system. The Block Diagram provides an overview of the system's modular structure, detailing the relationships among its key components, including the data acquisition module, feature extraction unit, classification model, and ITS integration layer. Lastly, the System Architecture serves as a blueprint for how hardware, software, and algorithmic components communicate and work together to achieve accurate detection of self-handicapping behaviors. It guides the integration, scalability, and maintainability of the entire system, ensuring that each module functions cohesively to meet the project's objectives.

Activity Diagram



Use Case Diagram

Class Diagram

System Architecture

3.3.Hardware and Software Specifications

Hardware:

Hardware	Minimum	Recommended
Processor	Intel 7th Gen Processor/Ryzen 5 Processor (6-core up)	Intel 9th Gen Processor/Ryzen 7

		Processor (8-core up)
Ram	8GB	16GB
Storage	5GB	50GB
Gpu	Integrated Graphics Card	NVIDIA GeForce 1050 RTX 2060

Table ?. Hardware Specifications for Training Data

Hardware Specification	
Processor	Intel 8th Gen Processor / Ryzen 5th Gen
GPU	Integrated Graphics Card
RAM	8gb
Hard Disk	At least 8gb of free space

Table ?. Hardware Specifications for Usage

Software:

Software Specification
Google Chrome or Any Stable Browser
Windows 10 or higher
Internet Connection

Table ?. Software Specifications for Training Data

Task	**Library / Repositories**
Convolutional Neural Network	Keras-model OpenCV
Recurrent Neural Network	PyTorch / Keras
Landmark Detection	

Table ?. Software Specifications for Usage

3.4.Method in Developing the Software

SCRUM

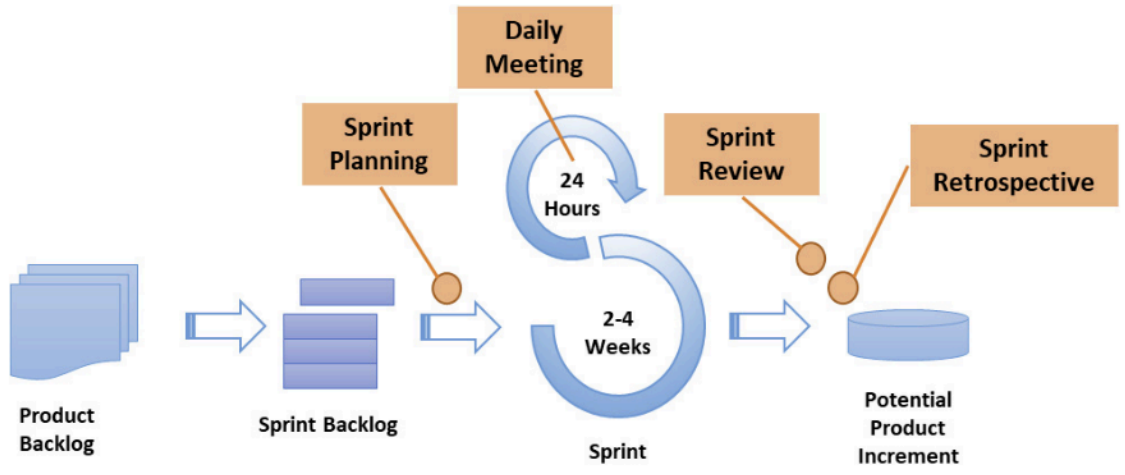


Figure ?. **Scrum life cycle**
Reprinted from
Zayat, W., & Senvar, O. (2020). Framework Study for Agile Software Development Via Scrum and Kanban. In International Journal of Innovation and Technology Management (Vol. 17, Issue 4). World Scientific. https://doi.org/10.1142/S0219877020300025

3.5.Method in Evaluating the Study

3.5.1 System Testing

3.5.2 White box Testing

3.5.3 Black box Testing

3.6.Data Gathering Procedure

Prototype Development
Data Collection

3.6.1 Survey Questionnaire

3.6.2 Local and Online Libraries

3.7.Respondents of the Study

3.8.Statistical Treatment of Data

3.8.1 Weighted Mean

3.8.2 Likert Scale

Table 3.8. Likert Scale

3.9.Dataset

3.10.Algorithm

Video Model:
Behavior model: Random Forest
Ensemble Model: Logistic Regression

3.11.Model Evaluation

Precision: The ratio of correctly predicted self-handicapping behaviors to the total predicted self-handicapping behaviors, indicating how precise the model is in identifying true behavioral instances.

$$precison = \frac{tp}{tp + fp}$$

Accuracy: The overall percentage of correctly classified behaviors, both self-handicapping and non-self-handicapping, against the total number of instances, reflecting the model's general performance in behavior detection.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

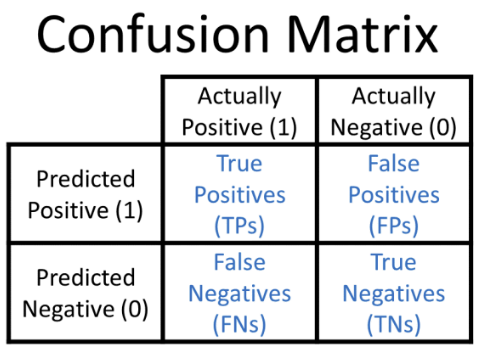
Recall: The ratio of correctly predicted self-handicapping behaviors to the total actual occurrences, measuring the model's capacity to identify all instances of self-handicapping behavior.

$$recall = \frac{tp}{tp + fn}$$

F1-Score: The harmonic mean of Precision and Recall, providing a balanced assessment of the model's performance when both false positives and false negatives are of concern.

$$F\alpha\text{-}score = \frac{(\alpha^2 + 1) \cdot precision \cdot recall}{\alpha^2 \cdot precision + recall}$$

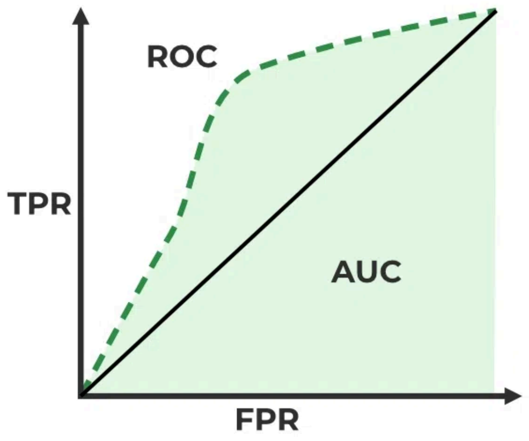
Confusion Matrix: A tabular representation that summarizes the model's classification performance by displaying the counts of True Positives, True Negatives, False Positives, and False Negatives. It provides a clear overview of how well the model distinguishes between self-handicapping and non-self-handicapping behaviors, allowing for detailed evaluation of prediction errors and model accuracy.



Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

AUC/ROC: The Area Under the Receiver Operating Characteristic Curve (AUC/ROC) measures the model's ability to distinguish between self-handicapping and non-self-handicapping behaviors across different classification thresholds. A higher AUC indicates better discriminative performance.



3.12.

Software Quality Evaluation(ISO 25010)

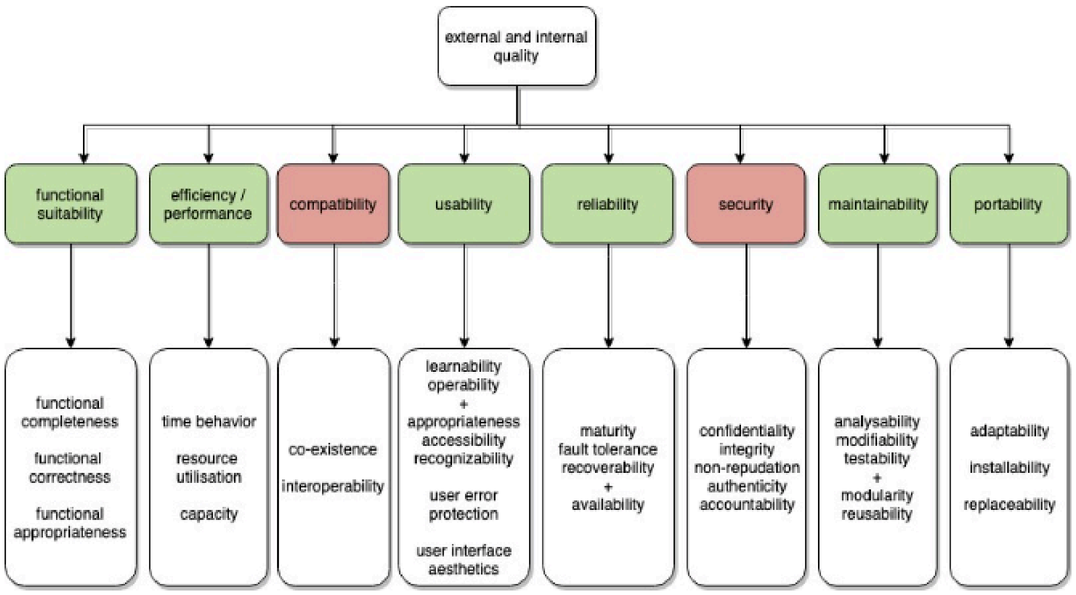


Figure ?. ***ISO/IEC 25010 categorization of software quality requirements***
Source: ISO20500.com
Reprinted From

Rebeś, P., & Rebeś, P. (n.d.). Software Quality Standards, How and Why we applied ISO 25010 | Monterail. Monterail. [*https://www.monterail.com/blog/software-qa-standards-iso-25010*](https://www.monterail.com/blog/software-qa-standards-iso-25010)

The ISO/IEC 25010 framework will be used to evaluate the system's overall software quality, focusing on key attributes such as functionality, reliability, usability, performance efficiency, maintainability, and security. This evaluation ensures that the Cognitive Strategy Recognition System meets established standards for robustness and user satisfaction. By assessing these quality characteristics, the study aims to verify that the system performs effectively under real academic conditions, supports accurate behavioral detection, and maintains a high level of usability for both students and educators.

User Acceptance (TAM)

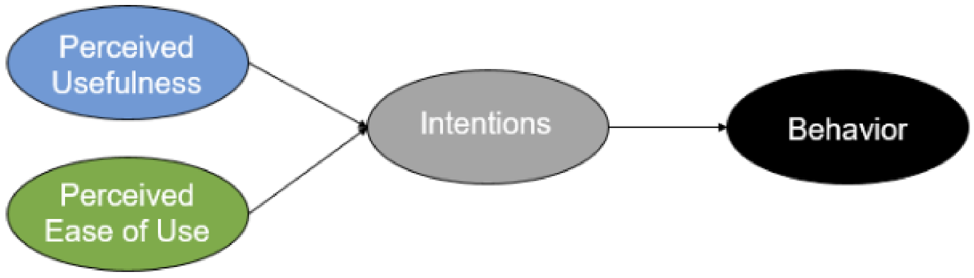


Figure ?. Technology Acceptance Model
Reprinted From
Worthington, A. K. (2021, May 30). Technology Acceptance model. Persuasion Theory in Action: An Open Educational Resource. [*https://ua.pressbooks.pub/persuasiontheoryinaction/chapter/technology-acceptance-model/*](https://ua.pressbooks.pub/persuasiontheoryinaction/chapter/technology-acceptance-model/)

The Technology Acceptance Model (TAM) will be utilized to evaluate the perceived usefulness and ease of use of the Cognitive Strategy Recognition System within the Intelligent Tutoring System environment. This framework helps assess how students and educators accept and interact with the system by measuring factors such as usability, functionality, and overall satisfaction. Feedback gathered through user testing and surveys will determine whether the system effectively supports learning and encourages positive engagement, ensuring its acceptance in real-world academic applications.

Chapter 4

PRESENTATION, ANALYSIS AND INTERPRETATION

This chapter contains detailed presentation and discussion of data analysis and the results of this study. The findings are generated from the survey questionnaires that is conducted to evaluate the system.

4.1 Presentation and Analysis of Software Evaluation

The survey questionnaire answered by the respondents are based from the ISO-9126 to

measure the software application.

4.2. Interpretation of Software Evaluation

4.2.1 Summary of Software Evaluation

4.2.2 Test Result of System Accuracy

Chapter 5

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

This chapter presents the summary of findings and conclusions based on the data analyzed from the survey questionnaire answered by the respondents and also the researchers will describe the recommendation for future researchers to improve the system.

Summary of Findings

Conclusions

Recommendations

BIBLIOGRAPHY

Adorico, M. A., Cesar, Iv A. A., Emma, V. S., Ruel, A. B., Nelia B. A., Mercelee P. P., (n.d.). The Living Conditions of University Students in Boarding Houses and Dormitories in Davao City, Philippines Retrieved from http://www.researchgate.net/publication/273709431_The_Living_Conditions_of_University_Students_in_Boarding_Houses_and_Dormitories_in_Davao_City_Philippines

Alan,Patrick ,W.Aswin , Natalia C., Sonya R. M. (2016)Development and evaluation of mobile application for room rental information with chat and push notification. Retrieved from http://jar.ssu.edu.ph/index.php/JAR/article/view/27

Milinovic., M. Maric., I. Cale., Z. Skiljan., D. Penezic., V. Rabljenovic., (2004).StuDOM project: local computer networks in student dormitories Retrieved fromhttps://ieeexplore.ieee.org/document/1372486?reload=true&arnumber=1372486

Nenny A., Andrew F., Miftahul F., (2017)Flow measurement of charges and electricity costs monitoring system with android based Iot (case study: Boarding house Adelina). Retrieved from http://jar.ssu.edu.ph/index.php/JAR/article/view/27

Qilin Y., Xi-a S., (2016) A research on the model of university apartment occupancy distribution based on the student preferences. Retrieved from https://ieeexplore.ieee.org/document/7854472

Rowee, Joy S. D. (2015). The Influence of Living Conditions of Boarding Houses and Dormitories on the Well-being of the State University Students Retrieved from https://ejournals.ph/article.php?id=11612

Santit N., Veera B., (2016)Selecting students to a dormitory using AHP Retrieved from https://ieeexplore.ieee.org/document/7748898

The Philippine Star. (2015).Lack of Dorms Retrieved from https://www.philstar.com/opinion/2015/08/10/1486739/lack-dorms

Erezo, S. L., Manaois, T. H., Maniacup, J. C., Punsalan, M. C., Valdez, S., & Cunanan, K. M. (2020). Academic self-handicapping strategies of students in a community college: A phenomenological study. PUNLA, City College of Angeles. Retrieved from https://cca.edu.ph/assets/images/Academic%20Self-Handicapping%20Strategies%20of%20Students%20in%20a%20Community%20College%20A%20Phenomenological%20Study%20.pdf

Bringula, R. P. (2019). Intelligent tutoring systems for Filipino learners: Current research, gaps, and opportunities. In Education Book Chapter. IGI Global.

https://www.igi-global.com/chapter/intelligent-tutoring-systems-for-filipino-learners/237246

Estrellado, C. J. P., & Miranda, J. C. (2023). Artificial intelligence in the Philippine educational context: Circumspection and future inquiries. International Journal of Scientific and Research Publications, 13(5). https://www.ijsrp.org/research-paper-0523/ijsrp-p13704.pdf

Go, M. A., Cesar, I. A. A., Emma, V. S., Ruel, A. B., & Nelia, B. A. (2020). Filipino teachers' compartmentalization ability, emotional intelligence, and teaching performance. ERIC. https://files.eric.ed.gov/fulltext/EJ1274296.pdf

Taduran, R. (2011). Mukha mo: A preliminary study on Filipino facial expressions. Social Science Diliman, 7(1), 1–26. https://journals.upd.edu.ph/index.php/socialsciencediliman/article/download/3605/3315

Endiape, J. M., & Hermosa, R. M. (2023). Academic self-handicapping and self-regulating learning strategies for student engagement in performance of Grade 8 students in Araling Panlipunan. Philippine EJournals. https://ejournals.ph/article.php?id=21844

Endiape, J. M., & Hermosa, R. M. (2023). Academic self-handicapping and self-regulating learning strategies for student engagement in performance of Grade 8 students in Araling Panlipunan. International Journal of Multidisciplinary: Applied Business and Education Research, 4(5), 123–134.

Abu-Naser, S. S. (2008). Developing an intelligent tutoring system for students learning to program in C++. International Journal of Computer Science and Information Security, 4(1), 1–7. https://www.researchgate.net/publication/26557161

Salloum, S. A., Alshurideh, M. T., & Shaalan, K. (2025). Emotion recognition for enhanced learning: Using AI to detect students' emotions and adjust teaching methods. Smart Learning Environments, 12(1). https://slejournal.springeropen.com/articles/10.1186/s40561-025-00374-5

Yıldırım, M., & Demir, A. (2019). Self-handicapping among university students: The role of procrastination, test anxiety, self-esteem, and self-compassion. Middle East Technical University. https://self-compassion.org/wp-content/uploads/2019/08/Barutc%CC%A7u-Y%C4%B1ld%C4%B1r%C4%B1m2019.pdf

Izadpanah, S., & Charmi, M. (2022). The effect of social networks on academic self-handicapping with the mediating role of self-regulatory learning strategies and academic achievement among EFL students. Frontiers in Psychology, 13, 987381. https://www.frontiersin.org/articles/10.3389/fpsyg.2022.987381/full

Kouam, J. (2024). The effectiveness of intelligent tutoring systems in supporting students with varying levels of programming experience. Discover Education, 2(1), 1–15. https://link.springer.com/article/10.1007/s44217-024-00385-3

Zhong, X., & Zhan, Z. (2025). An intelligent tutoring system for programming education based on informative tutoring feedback: System development, algorithm design, and empirical study. Interactive Technology and Smart Education, 22(1), 3–24. https://www.emerald.com/itse/article-abstract/22/1/3/1245798

Liu, Q., Jiang, X., & Jiang, R. (2025). Classroom Behavior Recognition Using Computer Vision: A Systematic review. Sensors, 25(2), 373. https://doi.org/10.3390/s25020373

Naveed, M. S. (2021). Comparison of C++ and Java in implementing introductory programming algorithms. Quaid-e-Awam University Research Journal of Engineering Science & Technology, 19(1), 95–103. https://doi.org/10.52584/qrj.1901.14

APPENDICES

APPENDIX A

THESIS PROPOSAL TITLE

APPENDIX B

SURVEY FORM

APPENDIX C

COMMUNICATION LETTER