Journal of Education and Teacher Training Innovation (JETTI)
Published by Pusmedia Group Indonesia

**Journal of Education and Teacher Training Innovation**
Volume 4 Issue 2 Year 2026 Pages 655 - 674
e–ISSN : 3025-3500
DOI: 10.61227
e-mail : jetti.pusmedia@gmail.com
https://journal.pusmedia.com/index.php/JETTI

# DISHCODE: Design of an Educational Game for Computational Thinking Using the Game Development Life Cycle Model

Alissa Isni Silviani Sutadi [1]*, Eka Fitrajaya Rahman [2], Erna Piantari [3]

[1,2,3] Department of Computer Science Education, Universitas Pendidikan Indonesia

Correspondence e-mail * : alissaisni@upi.edu

**Abstract:** The low level of computational thinking (CT) ability among Grade X Vocational School students in branching material is the primary problem underlying this research. This study aims to design an educational game named DISHCODE that systematically develops CT skills through structured branching programming material. The game targets Grade X PPLG students at SMKN 1 Sumedang, West Java, Indonesia. DISHCODE is developed as an extension of Kitchen Chaos, an open-source educational game developed by CodeMonkey, by adapting its culinary world theme and integrating branching code input as the core mechanic. The research employs a Research and Development (R&D) approach using the Game Development Life Cycle (GDLC) model, limited to the pre-production phase. The GDLC process produced five primary design artefacts, namely game mechanics and learning mechanics mapping, a Game Design Document (GDD), use case and activity diagrams, level mapping aligned with CT indicators, and a storyboard. The game comprises four levels of progressively increasing difficulty that map branching sub-topics, starting from single IF, sequential if-then, if-else, and nested if, onto four CT indicators covering decomposition, pattern recognition, abstraction, and algorithm design. Each level employs a distinct input mechanism calibrated to the cognitive demands of its target indicator. The design results indicate that the GDLC framework provides a systematic structure for translating CT-based learning objectives into coherent game design artefacts, and that DISHCODE's mechanic-indicator alignment positions it as a prospective structured medium for developing and reflecting students CT competency through gameplay, pending empirical validation in future implementation stages**.** These findings suggest that DISHCODE has the potential to offer teachers a ready-to-use game-based learning instrument that maps branching sub-topics directly onto CT indicators, enabling more structured and measurable CT instruction within vocational programming classes, and to provide learners with a progressively scaffolded gameplay experience that trains decomposition, pattern recognition, abstraction, and algorithm design through contextual branching problem-solving.

**Keyword :** branching, computational thinking, educational game, game development life cycle

**Article info:** Submitted : 2026-06-08 | Accepted : 2026-06-25 | Published : 2026-08-01



How to Cite :

## Introduction

Computational thinking (CT) is a 21st-century problem-solving competency. Wing (2006), credited with popularising the term, characterised CT broadly as a fundamental analytical skill comparable to reading, writing, and arithmetic, later

refining this into the thought processes involved in formulating problems and their solutions in a way that solutions can be carried out by an information-processing agent (Wing, 2010). In the Indonesian educational context, CT has been formally incorporated as a core competency in the Merdeka Curriculum under Permendikbud No. 37, reflecting institutional recognition of its centrality to informatics education (Zahid, 2020). Tabesh (2017) operationalised Wing's broad notion into a four-stage problem-solving process comprising decomposition, pattern recognition, abstraction, and algorithm design, tracing the framework to a constructivist lineage running from Piaget through Papert's (1980) argument that learning is strengthened when learners are engaged in constructing a meaningful product. To enact these four stages, Tabesh proposed a "*playground*" model, an accessible environment in which learners experiment numerically, geometrically, and procedurally, modelling and backtracking while searching for patterns, symmetry, and invariants rather than being instructed directly. This model is paired with a "use-modify-create" progression, in which learners are granted increasing ownership over open-ended tasks as their CT competency develops. This constructivist, playground-based conception of CT provides a direct theoretical warrant for adopting a game as the instructional medium in the present study, since a well-designed game can instantiate exactly such an experimental, learner-owned environment for practising the four CT stages.

Beyond these foundational definitions, subsequent scholarship has substantially elaborated the CT construct. Brennan & Resnick (2012) operationalised CT into three dimensions: computational concepts (e.g., sequences, loops, conditionals), computational practices (e.g., testing and debugging), and computational perspectives (e.g., expressing and connecting). Grover & Pea (2013) synthesised the K–12 CT literature, emphasising decomposition, abstraction, and generalisation as the core cognitive operations. Shute et al. (2017) further operationalised CT as a set of overlapping problem-solving skills with clear instructional implications, while Korkmaz et al. (2017) developed validated assessment instruments identifying creativity, algorithmic thinking, co-operation, critical thinking, and problem-solving as its measurable dimensions. This multi-dimensional perspective underscores the importance of designing educational tools that simultaneously address multiple CT indicators, which is the central design rationale of DISHCODE.

Despite this recognition, empirical evidence consistently shows that CT attainment among Indonesian vocational school students remains low. Nuraini et al. (2023) found that Grade X SMK students largely master only the decomposition indicator, while pattern recognition, abstraction, and algorithm design remain substantially below mastery level. Mardiah et al. (2023) similarly reported that 61.54% of SMK students fall in the low CT category, with persistent difficulty in constructing

systematic and logical solution pathways. This profile is particularly concerning at the SMK level, which directly prepares students for careers in information technology.

The subject domain most directly affected is branching material in structured programming. Misbah (2025) found through preliminary observation at SMKN 1 Cisarua that learning in the Fundamentals of Software and Game Development subject, particularly on branching structure material, remained heavily dominated by the teacher's active role while students tended to become passive listeners. This teacher-centred pattern reduced methodological variety, creating a monotonous learning atmosphere that lowered students' active participation and ultimately affected their learning outcomes and motivation toward branching material. Putra (2024) further identified several compounding difficulties in learning algorithms and programming at the SMK level: students tend to focus on memorising syntax rather than understanding logic, lack a foundational background in computing from prior education, and struggle to translate narrative problem scenarios into programming structures due to limited analytical and problem-solving skills. Interviews with an Informatics teacher at SMKN 1 Cimahi confirmed that branching is among the specific sub-topics where these difficulties are most prominently observed. This aligns with Prayoga et al. (2025), who identified that the central difficulty in branching is not syntactic but conceptual, namely translating a real-world conditional situation into a formally correct logical structure in code, a process that is structurally equivalent to performing all four CT operations simultaneously, making branching an exceptionally well-suited vehicle for CT development.

Educational games have been widely demonstrated to be effective interactive media for programming and CT learning. Saputra et al. (2024) developed a puzzle-genre educational game for HTML programming at SMK that achieved feasibility scores of 95% from media experts and 83% from content experts. Graceota et al. (2021) showed that game-based environments sustain learner attention more effectively than conventional methods by creating experientially distinct learning encounters. Sarifah et al. (2022) added that well-designed educational games sequence learning content with rich visual feedback, sustaining engagement and reinforcing understanding progressively. Wibawanto (2020) further noted that educational games improve student mastery of a subject through embedded and consequential practice.

A relevant starting point for the present work is Kitchen Chaos, an open-source educational game developed by CodeMonkey that simulates a culinary kitchen environment. Kitchen Chaos employs a task-completion mechanic in which players execute sequential cooking operations within a time constraint. While Kitchen Chaos was not originally designed as a CT or programming learning tool, its culinary world structure, order-based task system, and immediate feedback mechanics provide a compelling thematic and structural foundation for a game targeting branching programming. The present study therefore develops DISHCODE as an extension of

Kitchen Chaos, preserving its culinary theme and feedback architecture while replacing its original mechanics with branching code input as the driver of gameplay, and explicitly mapping each game level to a specific CT indicator.

Prior studies on game-based CT development have produced promising results but leave important gaps unaddressed. Rizky (2025) demonstrated that problem-based learning improved CT in networking subjects but did not isolate gameplay to a specific CT sub-competency or branching material. Sari & Laksana (2026) reported a 22% average CT improvement using Blockly Games but limited their study to elementary-level basic coding. Saniyah et al. (2025) applied educational game media combining Wordwall and Quizizz in a Classroom Action Research and found progressive learning outcome improvements across two cycles, confirming that game-based approaches increase student engagement and cognitive achievement, however the study focused on science content at the elementary level and did not address CT indicators or structured game design frameworks. Irfan et al. (2025) developed "Petualangan Si Wunda" using the ADDIE model on Roblox Studio and explicitly mapped all four CT indicators onto game missions achieving a high N-Gain score of 0.77, yet their work targeted Grade IV elementary students on science content, leaving CT-integrated game design for secondary vocational programming content unexplored. No prior work has systematically mapped all four CT indicators onto a progression of branching sub-topics within a single game developed using the GDLC framework. The present study addresses this gap by designing DISHCODE, a GDLC-developed educational game that targets all four CT indicators through progressive branching programming levels at the Grade X SMK level.

## Methodology

This research employs a Research and Development (R&D) approach using the Game Development Life Cycle (GDLC) model as the primary framework for game production. The GDLC model was selected because it provides a structured, iterative workflow specifically designed for game development, ensuring that design decisions are organised into clearly bounded and reviewable phases.

GDLC berisi siklus khusus yang terdapat pada pengembangan *game* dan memiliki tujuan untuk menjawab tiga pertanyaan yang hadir ketika mengembangkan *game*: kriteria seperti apa yang harus dipertimbangkan ketika mengembangkan *game*, bagaimana menciptakan *game* dengan kualitas yang bagus, dan langkah apa saja yang harus dilakukan ketika mengembangkan *game* (Luthfan & Mahardhika, 2021).

The GDLC model used in this study follows the framework proposed by (Ramadan & Widyani, 2013), which defines game production as a cyclical process comprising Initiation, Pre-Production, Production, Testing, Beta, and Release phases. This framework has been widely adopted in educational game development research

(Nurcholis et al., 2021; Saputra et al., 2024) because its iterative review cycles are well-suited to aligning game design decisions with learning objectives.

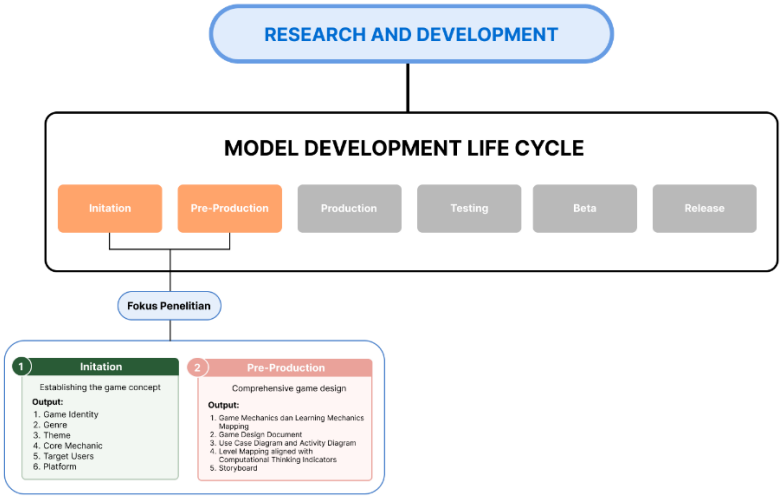


**Figure 1. GDLC Research Framework and Phase Scope**

The research is limited to the pre-production phase of GDLC, which comprises two sub-phases: Initiation and Pre-Production. The Initiation phase establishes the fundamental game concept, including game identity, genre, theme, core mechanics, target users, and platform. The Pre-Production phase translates that concept into a full set of design artefacts.

The GDLC framework was combined with computational thinking-based instructional design principles to ensure that every game design decision always has a clear learning objective, rather than merely producing a visually appealing game. The four CT indicators defined by Wing (2006) and further elaborated by Ansori (2020), namely decomposition, pattern recognition, abstraction, and algorithm design, served as benchmarks in designing the mechanics and tasks at each level, so that each level is not designed arbitrarily but refers to the specific CT indicator it aims to develop. The strength of this study lies in combining both approaches simultaneously, where the GDLC ensures a systematic game production process while the CT indicators ensure that every game element has a measurable learning function.

Because this study was restricted to the pre-production phase of the GDLC model, the outputs are limited to design artefacts, including game mechanics, learning mechanics, Game Design Documents (GDD), use case and activity diagrams, level mapping, and storyboards. Consequently, expert validation, user testing, implementation, and effectiveness evaluation were not conducted within the scope of this research. Therefore, the findings should be interpreted as design-stage outcomes rather than evidence of implementation effectiveness. This limitation is acknowledged as a methodological constraint of the study.

### Research Context

The target users of DISHCODE are Grade X students of the Software and Game Development (PPLG) concentration at SMKN 1 Sumedang, West Java, Indonesia. This population was selected because branching material is a core topic in their Structured Programming curriculum under Phase E of the Merdeka Framework, and because prior diagnostic data from the school indicated low CT performance on branching tasks. The game is designed for individual desktop use on Windows-based PCs, consistent with the school's available hardware infrastructure.

### Data Collection Process

The design process followed the two GDLC phases sequentially. In the Initiation phase, the core game concept was defined through analysis of the CT and branching learning objectives, review of the Kitchen Chaos architecture, and specification of the game's identity parameters. In the Pre-Production phase, five design artefacts were produced iteratively: the mechanics-learning mechanics mapping, the GDD, use case and activity diagrams, level mapping with CT indicators, and the storyboard. Each artefact was reviewed against the CT benchmarks to verify alignment before the next artefact was developed. The design process was qualitatively documented and analysed through descriptive analysis of each artefact's structure, content, and pedagogical rationale.

## Result and Discussion

### Finding

#### 1. Initiation Phase: Core Game Concept

The Initiation phase established the identity and foundational concept of DISHCODE. The game name combines 'dish,' referencing the culinary theme inherited from Kitchen Chaos, and 'code,' representing branching programming as the core mechanic. Table 1 summarises the core concept parameters established in this phase.

**Table 1. Core Concept of DISHCODE**

| Aspect | Description |
|---|---|
| Game Title | DISHCODE |
| Genre | Serious Game |
| Theme | Culinary world and cooking kitchen (adapted from Kitchen Chaos, CodeMonkey) |
| Core Mechanic | Branching code input as the driver of game flow |
| Target Users | Grade X PPLG students, SMKN 1 Sumedang |
| Platform | PC / Desktop (Windows) |
| Game Engine | Unity |
| Learning Material | Branching (if, if-else, nested if) – Structured Programming |
| Number of Levels | 4 levels with progressively increasing complexity |

DISHCODE was designed as a direct extension of Kitchen Chaos by CodeMonkey. Kitchen Chaos provides an open-source culinary simulation framework

featuring a kitchen environment, a chef character, order-based tasks, and immediate feedback on task outcomes. In DISHCODE, this structural framework is preserved but fundamentally repurposed: the original movement and action mechanics of Kitchen Chaos are replaced by branching code input mechanics, so that the player's character can only act correctly when the student constructs a logically sound branching statement. The culinary world context is retained because it provides a familiar and accessible problem scenario that maps naturally onto real-world conditional situations (e.g., 'IF the ingredient is available, THEN add it'), reducing the cognitive distance between everyday logic and formal programming constructs.

## 2. Pre-Production Phase: Design Artefacts

### a. Game Mechanics and Learning Mechanics Mapping

Game mechanics and learning mechanics were defined simultaneously to ensure every game element serves a clear learning function. Table 2 presents the mapping between game mechanics and their corresponding learning mechanics.

**Table 2. Mapping of Game Mechanics and Learning Mechanics**

| Game Mechanic | Learning Mechanic |
|---|---|
| Levels (sequential, locked) | Action/Task |
| Movement (code as movement key) | Simulation, Responsibility |
| Roleplay (chef in culinary world) | Discover, Explore |
| Strategy/Planning (construct logic before executing) | Analyze, Observation |
| Simulate (contextual scenarios) | Simulation, Experimentation |
| Quick Feedback (instant correct/incorrect) | Feedback, Responsibility |
| Rewards/Penalties (score, lives) | Feedback, Action/Task |
| Tutorial (initial and between-level) | Tutorial |

The most structurally significant mechanic is the movement mechanic, in which the branching code entered by the student directly drives character movement and task execution. This design ensures that correct algorithm design produces tangible in-game consequences, making the abstract process of constructing branching logic experientially concrete. Saputra et al. (2024) similarly found that consequence-driven

game mechanics substantially improve students' grasp of logical flow compared to passive instruction.

**b. Game Design Document (GDD)**

The GDD was compiled as the central technical and pedagogical reference document. Table 3 summarises the overall game system, Table 4 presents the game rules, Table 5 describes the level mechanics, and Table 6 details the reward and penalty system.

**Table 3. Game System Summary**

| System | Description |
|---|---|
| Level System | Access to the next level is controlled by the teacher, who manually unlocks it in accordance with the learning sequence. |
| Lives System | Players in Levels 2–4 begin with 5 lives. Each incorrect answer deducts 1 life. Game Over when lives reach 0. |
| Timer System | Level 1 has a 4-minute time limit. Time Over screen is displayed if time expires before level completion. |
| Score System | Scores are calculated based on answer accuracy. Each correct answer adds points; incorrect answers do not deduct scores (penalty is via life deduction). |
| Quick Feedback | Instant text messages appear with every correct or incorrect answer. |
| Guidebook | A branching material reference accessible at any time during play without penalty, designed to reduce cognitive load on syntax recall. |

The game system in the GDD was designed on the basis of the game mechanics and learning mechanics established in the preceding stage. The sequential locked-level system reflects the levels mechanic while simultaneously activating the action/task learning mechanic, whereby students can only proceed to the next level after genuinely mastering the material of the previous one. The lives and score systems function as diagnostic indicators that directly reflect the quality of each student's algorithmic decision-making through game progress. The Guidebook, accessible at any time during play, is designed to reduce students' cognitive load in recalling syntax, so that their thinking capacity can be redirected entirely toward branching logic reasoning. The game rules were established to ensure consistency of the playing experience while

preserving the integrity of the learning process. The complete set of game rules in DISHCODE is summarised in Table 4.

**Table 4. Game Rules**

| No | Game Rule |
|---|---|
| 1 | In Level 1, players have 4 minutes and no lives. In Levels 2–4, there is no time limit but players have 5 lives. |
| 2 | Each incorrect answer or logic arrangement deducts 1 life. |
| 3 | The next level can only be accessed after the teacher manually unlocks it. |
| 4 | If lives reach 0, the Game Over screen is displayed; the player may retry. |
| 5 | If time expires, the Time Over screen is displayed; the player may retry. |
| 6 | The Guidebook may be opened at any time without penalty. |
| 7 | Students may only press RUN after all answer inputs or logic blocks have been filled. |

As shown in Table 5, each level features a different input mechanism corresponding to the complexity of the material. Level 1 requires a simple Yes/No choice; Level 2 demands step-by-step pseudocode input; Level 3 asks students to select conditions and actions via dropdowns; and Level 4 grants full autonomy to assemble logic blocks hierarchically in a workspace. This pattern demonstrates that as the level increases, student involvement in designing solutions grows progressively. The explicit mapping of each level to its corresponding CT indicators is presented in Table 7.

**Table 5. Level Mechanism Descriptions**

| Lvl | Branching Material | Input Mechanism |
|---|---|---|
| 1 | Single IF | Yes/No button selection based on displayed ingredient condition |
| 2 | Sequential If-Then | Step-by-step pseudocode input following burger recipe sequence |
| 3 | If-Else (two-way) | Dropdown selection of condition, IF action, and ELSE action |
| 4 | Nested If | Free-form logic block assembly in empty workspace |

In addition, the reward and penalty system is designed to deliver immediate and proportional feedback for every player action. Rewards are provided as positive reinforcement for correct answers, while penalties serve as logical consequences for

errors that prompt students to reflect on and refine their reasoning. The specific conditions and system responses for each reward and penalty are presented in Table 6 below.

**Table 6. Reward and Penalty System**

| Category | Condition | System Response |
|---|---|---|
| Reward | Correct answer or logic arrangement | Score increases + positive feedback displayed instantly |
| Reward | Completing one order | Order count increases; gameplay continues to next order |
| Reward | Completing all orders in a level | Level complete; score summary displayed |
| Penalty | Incorrect answer or logic arrangement | Life deducted by 1; corrective feedback displayed |
| Penalty | Lives exhausted (0) | Game Over screen; player may retry |
| Penalty | Time expired (Level 1) | Time Over screen; player may retry |

The Guidebook system is a deliberate cognitive load management feature. By making branching syntax references available at any time without penalty, DISHCODE ensures that students' limited working memory capacity is directed entirely toward logical reasoning rather than syntax recall. This aligns with Silvia et al. (2023) finding that CT development requires deliberate instructional scaffolding that progressively increases cognitive demand while managing load.

**c. Use Case and Activity Diagrams**

The use case diagram of DISHCODE illustrates all interactions available to a single primary actor: the player. There are four main use case groups: Enter Game, Play Game, View Tutorial, and Exit Game. Within Play Game, an <<include>> relationship connects to the Play Level sub-use case, which branches into Select Level, Continue Level, and four level options. Play Level has <<extend>> relationships to three conditionally triggered use cases: Display Score, Receive Feedback, and Receive Hint. All use-case-to-use-case relationships are represented by dashed arrows with UML <<include>> or <<extend>>. The three colour groups in the diagram indicate: blue for core mandatory interactions accessible from the main menu; green for sub-

interactions within the Play Game flow; and orange for conditionally triggered extend relationships.

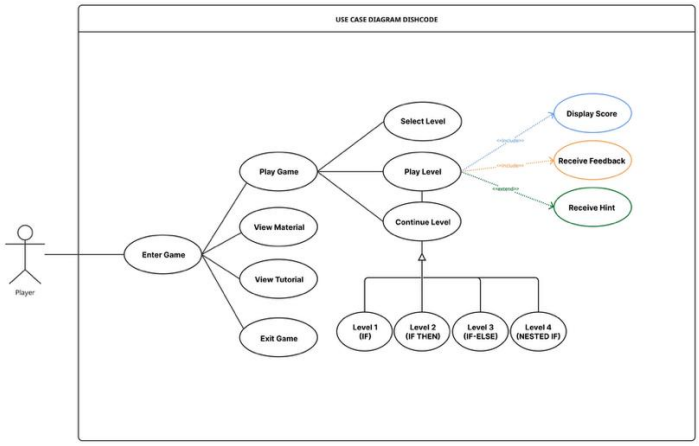


**Figure 2. Use Case Diagram of DISHCODE**

Activity diagrams were developed for each of the four levels to describe the sequential flow of player actions and system responses during gameplay. Each diagram illustrates how the interaction cycle works from the moment a student receives a problem scenario to the point where the system provides feedback on the student's answer.

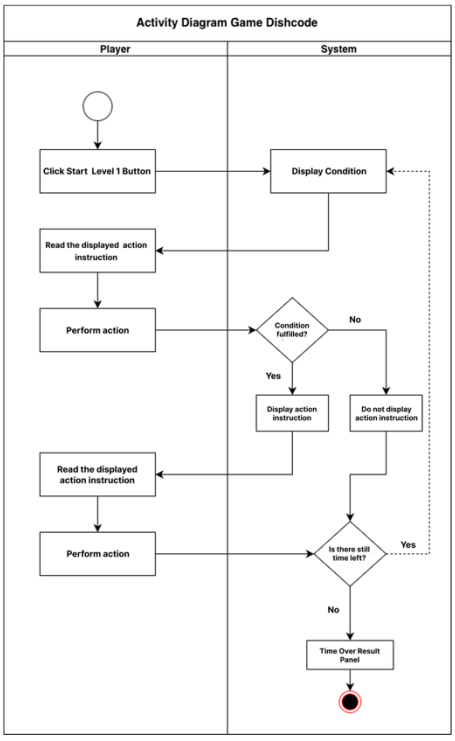


**Figure 3. Activity Diagram Level 1**

The activity diagram for Level 1 illustrates a repeating condition-response cycle driven by a time constraint. The system displays an ingredient condition, the student selects Yes or No, and the system immediately validates the input and displays an

action instruction if the condition is met. This cycle repeats until the four-minute timer expires. The simplicity of the Yes/No input mechanism isolates the most fundamental CT operation at this level: decomposition of a single binary condition, allowing students to focus entirely on identifying whether a given condition is true or false without being burdened by syntax construction.

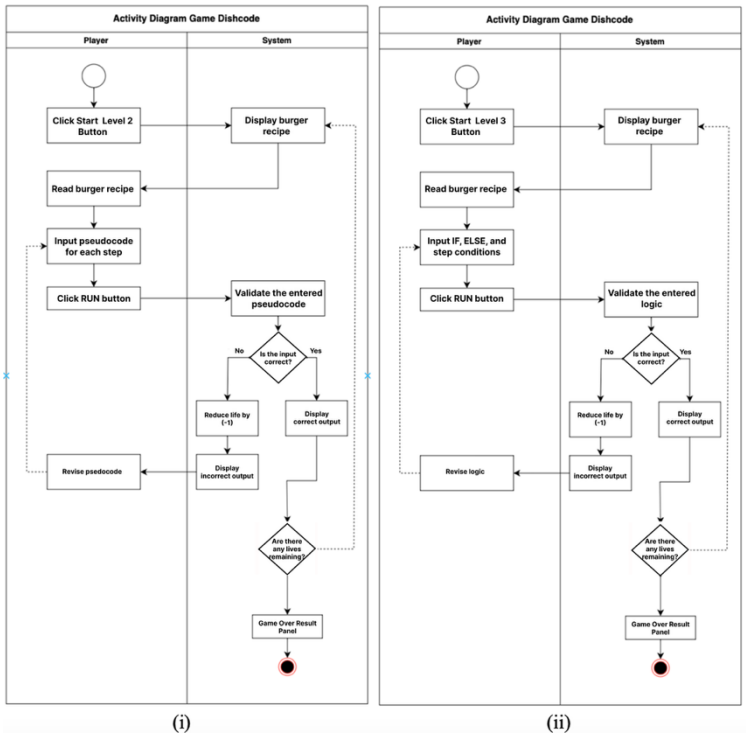


**Figure 4. Activity Diagrams Level 2 (left) and Level 3 (right)**

The activity diagram for Level 2 shows a step-by-step pseudocode input flow structured around a burger recipe sequence. The system presents an order, the student fills in each pseudocode line one step at a time following the recipe procedure, and the system validates the completed sequence before proceeding to the next order. If the sequence is incorrect, the system provides a hint. This flow trains pattern recognition by requiring students to identify the repeating condition-action structure of a recipe, and algorithm design by requiring them to translate that structure into a correctly ordered pseudocode sequence. The activity diagram for Level 3 introduces a two-way decision flow. The system presents a cooking scenario with two possible outcomes, and the student selects the condition, the IF action, and the ELSE action from three separate dropdown menus before pressing RUN. The system validates the complete arrangement and provides a warning with a correction opportunity if any selection is

incorrect. This flow demands abstraction, as students must simultaneously model both branches of a conditional structure rather than focusing on a single execution path.

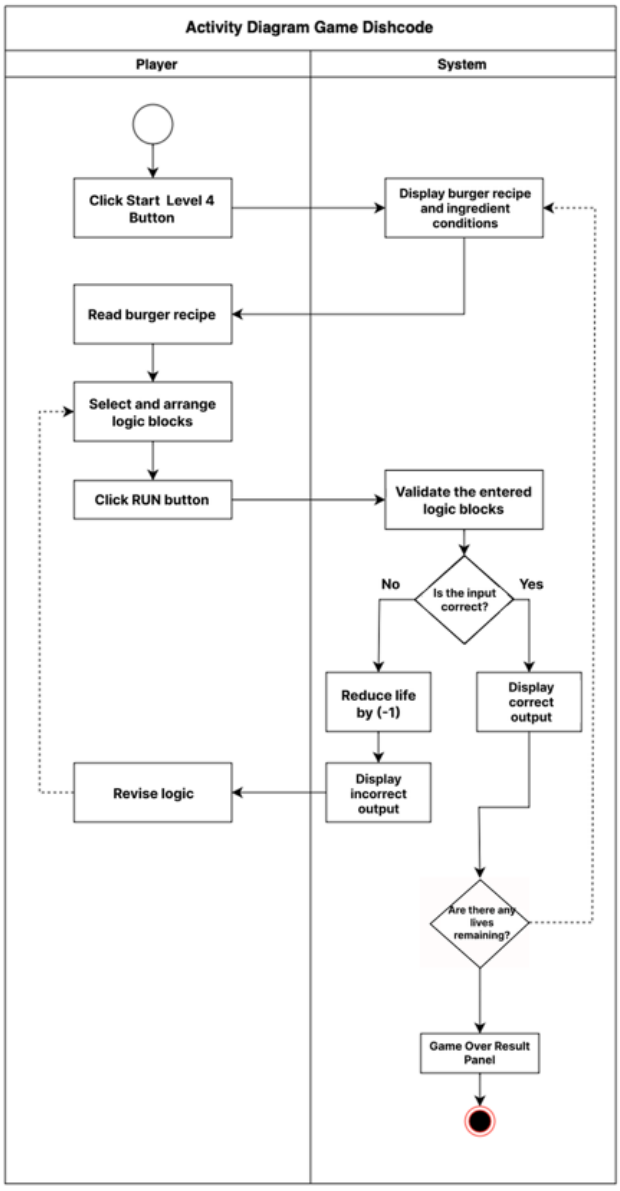


**Figure 5. Activity Diagram Level 4**

The activity diagram for Level 4 presents the most cognitively demanding interaction flow in the game. The system displays a complex multi-ingredient scenario, and the student independently assembles a complete nested if structure by selecting and arranging IF blocks, conditions, and actions in an empty workspace. After pressing RUN, the system validates the entire block arrangement and displays the execution output, leaving the student to interpret the result and make corrections independently. This fully open-ended flow activates all four CT indicators simultaneously, as students must decompose the problem, recognise nested patterns

within it, abstract away irrelevant details, and design a hierarchically correct algorithm from scratch.

Based on the activity diagrams above, all four levels share the same core interaction cycle: the system presents a branching scenario, the student constructs a logical response through the designated input mechanism, the system validates the input, and immediate feedback is delivered. What distinguishes each level is the nature of the input step, which grows progressively in cognitive demand from a simple Yes/No selection in Level 1 to independent free-form logic block assembly in Level 4. This escalating input complexity is deliberately aligned with the CT indicators targeted at each level, ensuring that the interaction flow itself functions as a scaffolded CT training mechanism.

**d. Level Mapping with CT Indicators**

The four levels are designed with progressively increasing cognitive complexity. Table 7 presents the explicit mapping of each level to its corresponding CT indicators and game context.

**Table 7. Level Mapping with CT Indicators**

| Level | Branching Material | CT Indicator(s) | Game Context |
|---|---|---|---|
| 1 | Single IF (Yes/No) | Decomposition, Pattern Recognition | Ingredient availability condition scenarios in a kitchen prep task |
| 2 | Sequential If-Then (Pseudocode) | Pattern Recognition, Algorithm Design | Burger-making recipe as a sequential algorithm |
| 3 | If-Else (two-way branching) | Abstraction, Algorithm Design | Two-way cooking decision with conditional outcomes |
| 4 | Nested If (multi-level) | Decomposition, Pattern Recognition, Abstraction, Algorithm Design | Complex multi-ingredient condition requiring hierarchical logic |

The progression from Level 1 to Level 4 mirrors the scaffolding approach recommended by Silvia et al. (2023), in which CT development requires incremental increases in cognitive demand. Level 1 isolates decomposition and pattern recognition at their most basic form: a single binary condition. Level 2 extends to sequential algorithm design through multi-step pseudocode. Level 3 introduces abstraction by requiring students to simultaneously model both branches of a two-way condition.

Level 4 activates all four CT indicators simultaneously through nested if construction, representing the highest level of branching complexity in the SMK curriculum.

**e. Storyboard**

The storyboard describes the visual layout and interaction flow of nine game scenes: Main Menu, Tutorial Menu, Material Menu, Level 1, Level 2, Level 3, Level 4, Game Over, and Time Over.

| Nama Scene | Gambar | Deskripsi | Aksi User |
|---|---|---|---|
| **Main Menu** |  | The initial screen of the game. Displays the DISHCODE logo and chef character. Four navigation buttons are available: Start, Materials, Tutorial, and Exit. | Press one of the navigation buttons as needed. |
| **Tutorial Menu** |  | A gameplay guide page containing five tutorial sub-menus. The content is presented sequentially using Next and Back buttons. A chef character serves as a visual guide. | Select a sub-menu, navigate between pages, and close the menu using the X button to return to the Main Menu. |
| **Material Menu** |  | A learning page containing structured branching materials. Each sub-topic includes explanations and flowchart illustrations. | Select a sub-topic, read the material and diagrams, navigate between pages, and return to the Main Menu. |
| **Level 1** |  | The MESSAGE panel displays ingredient conditions with Yes/Nobuttons. Includes a kitchen background, an OUTPUT panel showing action instructions, a NOTES panel, and GUIDEBOOK and EXITbuttons. | Read the condition, select Yes or No, review the output, and use the Guidebook if necessary. |
| **Level 2** |  | The RECIPE panel displays burger-making steps, a PSEUDOCODE input field, an OUTPUT panel, and RUN, GUIDEBOOK, and EXIT buttons. | Read the recipe, enter pseudocode step by step, press RUN, and review the |

| | | | |
|---|---|---|---|
| | | | output and hints. |
| **Level 3** |  | The INSTRUCTION panel presents a scenario to solve. The LOGIC area contains three dropdown rows for the condition, IF action, and ELSE action. Also includes an OUTPUT panel and RUN, GUIDEBOOK, and EXIT buttons. | Select the condition, IF action, and ELSE action from the dropdown menus, press RUN, and review the output. |
| **Level 4** |  | The INSTRUCTION panel displays a list of ingredient conditions. Includes an empty logic-block workspace, a LOGIC BLOCKSpanel containing ready-to-use blocks, and RUN and GUIDEBOOKbuttons. | Arrange logic blocks hierarchically in the workspace, press RUN, and revise the arrangement based on the output. |
| **Game Over** |  | Displayed when all lives are depleted. Shows a motivational message, score summary, number of completed orders, and remaining time. | Press Retry to replay the level or Main Menu to return to the beginning. |

Each level scene consistently provides an instruction panel, an answer workspace, a real-time output panel, and navigation buttons. The Guidebook button is present and accessible at every level. This interface consistency ensures students do not need to adapt to a new layout at each level, preserving cognitive resources for logical reasoning tasks. The culinary world visual language kitchen backdrops, chef character, ingredient panels is directly adapted from Kitchen Chaos and maintained uniformly across all scenes to sustain thematic coherence.

## Discussion

The design process suggests that the GDLC framework provides a structure basis for translating CT learning objectives into coherent game design artefacts, though empirical validation remains necessary to confirm its effectiveness. The two-phase GDLC process (Initiation and Pre-Production) produced a complete and internally consistent design package in which every element mechanics, levels,

rules, feedback systems, and visual layout can be traced back to a specific CT indicator. This traceability is a structural advantage over less systematic design approaches.

The development of DISHCODE from Kitchen Chaos represents a purposeful adaptation strategy. Rather than designing a game from scratch, this approach leverages an existing, field-tested culinary simulation framework and repurposes it for CT-targeted programming education. The Kitchen Chaos architecture provides a structurally sound base: its order-completion loop maps naturally onto the input-validation-feedback cycle required for CT practice, and its culinary world context provides an intuitive real-world analogy for conditional logic. By replacing Kitchen Chaos's original action mechanics with branching code input, DISHCODE transforms a recreational simulation into a CT development tool without sacrificing the engagement properties that make the original game effective. This development strategy aligns with Graceota et al. (2021) finding that games sustain learner attention precisely because they create experientially distinct and personally engaging learning environments.

DISHCODE's design can be positioned more clearly when compared with existing CT-focused educational games. Irfan et al. (2025) reported a 22% average CT improvement using Blockly Games, however their study remained limited to elementary-level basic coding and did not differentiate game mechanics across distinct CT indicators or organise content around a progressive sub-topic structure. Irfan et al. (2025) "Petualangan Si Wunda" represents the closest precedent in terms of explicit CT indicator mapping, as it assigns each game mission to a specific CT component within an ADDIE-developed game, yet it operates through narrative exploration rather than code input mechanics and targets elementary science content rather than vocational programming. DISHCODE distinguishes itself from both by combining explicit four-indicator mapping, progressive branching sub-topic structure, and code input as the core mechanic within a single coherent game developed under the GDLC framework, making it the first design to address all these dimensions simultaneously at the SMK level.

The mechanic-indicator alignment documented in Table 2 and the level-indicator mapping in Table 7 together ensure that DISHCODE is not merely a thematically game-like learning tool but a structurally calibrated CT instrument. Each level's input mechanism is specifically designed to demand the cognitive operations corresponding to its target CT indicators: Yes/No selection demands decomposition of a binary condition; pseudocode input demands sequential algorithm construction; dropdown selection demands simultaneous abstraction of both conditional branches; and free-form block assembly demands full four-indicator integration. This progressive mechanic design is consistent with Silvia et al. (2023) scaffolding principle

and with Saputra et al. (2024) finding that consequence-driven mechanics improve logical reasoning more effectively than passive instruction.

The Guidebook feature addresses a key instructional design challenge in programming education: the cognitive interference of syntax recall. By providing an always-accessible reference without penalty, DISHCODE separates the cognitive task of logic construction from the task of syntax memorisation, enabling students to focus their working memory on CT reasoning. This design choice reflects Ansori (2020) characterisation of CT as fundamentally a logical reasoning competency rather than a syntax knowledge competency.

The present study is limited to the design phase. Empirical validation through expert review, usability testing, and CT pre-post assessment is required to confirm the game's effectiveness and reliability as a learning instrument. Future research should implement the full GDLC production cycle including prototype development, alpha and beta testing, and post-release evaluation to provide evidence of learning outcomes.

## Conclusion

This study produced the design of DISHCODE, an educational game developed using the Game Development Life Cycle (GDLC) model to support computational thinking (CT) development in branching programming material for Grade X SMK students. DISHCODE was developed as a purposeful extension of Kitchen Chaos by CodeMonkey, preserving its culinary world architecture and feedback mechanics while replacing the original gameplay with branching code input as the core mechanic. The GDLC pre-production phase yielded five primary design artefacts: game mechanics and learning mechanics mapping, a Game Design Document, use case and activity diagrams, level mapping with CT indicators, and a storyboard. The game comprises four progressively complex levels that map branching sub-topics single IF, sequential if-then, if-else, and nested if onto decomposition, pattern recognition, abstraction, and algorithm design. The input mechanism at each level is specifically calibrated to the CT demands of its target indicator, and the Guidebook system manages cognitive load by separating syntax recall from logical reasoning. The GDLC framework shows promise as a structured production methodology for CT oriented educational game design, pending empirical validation. Future research should proceed to prototype development, expert validation, usability testing, and empirical CT assessment to confirm DISHCODE's effectiveness as a learning and evaluation instrument.

## References


Ansori, M. (2020). Pemikiran Komputasi (Computational Thinking) dalam Pemecahan Masalah. *DIRASAH*, *3*(1). https://ejournal.iaifa.ac.id/index.php/dirasah

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *AERA* (Vol. 2012).

Graceota, A., Budiyono, & Slamet, I. (2021). Mathematics Game as Interactive Learning Media In COVID-19 Pandemic Era. *Journal of Physics: Conference Series*, *1808*(1), 012041. https://doi.org/10.1088/1742-6596/1808/1/012041

Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. In *Educational Researcher* (Vol. 42, Number 1, pp. 38–43). https://doi.org/10.3102/0013189X12463051

Irfan, A., Yarmi, G., & Ramdhani, S. (2025). PENGEMBANGAN GAME EDUKASI PETUALANGAN SI WUNDA UNTUK MENINGKATKAN KEMAMPUAN BERPIKIR KOMPUTASIONAL (COMPUTATIONAL THINKING) SISWA KELAS IV SD. *Jurnal Muara Pendidikan*, *10*(2), 516–526. https://doi.org/10.52060/mp.v10i2.3756

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, *72*, 558–569. https://doi.org/10.1016/j.chb.2017.01.005

Mardiah, A., Yuliana Fitri, D., & Sains Dan Teknologi, F. (2023). Analisis Kemampuan Computational Thinking Siswa Pada Materi Sistem Persamaan Linear Tiga Variabel. In *J-PiMat* (Vol. 5, Number 2).

Misbah, I. (2025). *Pengembangan Media Pembelajaran dengan Integrasi Live Coding dan Model Pembelajaran AIR (Auditory, Intellectually, Repetition) untuk Meningkatkan Hasil dan Motivasi Belajar Siswa*. Universitas Pendidikan Indonesia.

Nuraini, F., Agustiani, N., & Mulyanti, Y. (2023). Analisis Kemampuan Berpikir Komputasi Ditinjau dari Kemandirian Belajar Siswa Kelas X SMK. *Jurnal Cendekia : Jurnal Pendidikan Matematika*, *7*(3), 3067–3082. https://doi.org/10.31004/cendekia.v7i3.2672

Prayoga, A. A., Afirianto, T., & Wardhono, W. S. (2025). *Hubungan Level Penyelesaian CodeCombat dengan Kemampuan Pemecahan Masalah Siswa pada Materi Pemrograman di SMA* (Vol. 9, Number 7). http://j-ptiik.ub.ac.id

Putra, J. A. (2024). *Rancang Bangun Multimedia Interaktif dengan Menerapkan Model Problem Based Learning untuk Meningkatkan Problem Solving Siswa SMK*. Universitas Pendidikan Indonesia.

Ramadan, R., & Widyani, Y. (2013). Game development life cycle guidelines. *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 95–100. https://doi.org/10.1109/ICACSIS.2013.6761558

Rizky, K. M. (2025). *Pengembangan Media Pembelajaran dengan Integrasi Live Coding dan Model Pembelajaran AIR (Auditory, Intellectually, Repetition) untuk Meningkatkan*

*Hasil dan Motivasi Belajar Siswa* [Universitas Pendidikan Indonesia]. https://repository.upi.edu/31506/

Saniyah, S., Guru, P., Dasar, S., Surabaya, U. N., & Info, A. (2025). PENERAPAN MEDIA GAME EDUKASI UNTUK MENINGKATKAN HASIL BELAJAR PADA MATERI SISTEM TATA SURYA. *JURNAL PENELITIAN PENDIDIKAN GURU SEKOLAH DASAR (JPPGSD)*, *13*(8), 2212–2220.

Saputra, M. D., Risnasari, M., Ningsih, P. R., Effindi, M. A., & Aini, N. (2024). Game Edukasi Berbasis Android Genre Puzzle Menggunakan GDevelop untuk Pembelajaran HTML di SMK Negeri 1 Kamal. *Journal of Education and Informatics Research*.

Sari, V. K., & Laksana, D. N. L. (2026). Penerapan Problem Based Learning Berbantuan Blockly Games Untuk Meningkatkan Kemampuan Berpikir Komputasional Siswa Sekolah Dasar. *Jurnal Pendidikan Dasar Flobamorata*, *7*(1), 18–25. https://doi.org/10.51494/jpdf.v7i1.2818

Sarifah, I., Rohmaniar, A., Marini, A., Sagita, J., Nuraini, S., Safitri, D., Maksum, A., Suntari, Y., & Sudrajat, A. (2022). Development of Android Based Educational Games to Enhance Elementary School Student Interests in Learning Mathematics. *International Journal of Interactive Mobile Technologies (IJIM)*, *16*(18), 149–161. https://doi.org/10.3991/ijim.v16i18.32949

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). *Demystifying Computational Thinking*.

Silvia, R. D., Pramasdyahsari, A. S., & Nizaruddin, N. (2023). ANALISIS KEMAMPUAN COMPUTATIONAL THINKING SISWA PADA MATERI ALJABAR DITINJAU DARI PEMECAHAN MASALAH MATEMATIS. In *Jurnal Pendidikan dan Riset Matematika* (Vol. 5, Number 2). http://ejurnal.budiutomomalang.ac.id/index.php/prismatika

Tabesh, Y. (2017). Computational thinking: A 21st century skill. *Olympiads in Informatics*, *11*(Special Issue), 65–70. https://doi.org/10.15388/ioi.2017.special.10

Wibawanto, W. (2020). *GAME EDUKASI RPG (ROLE PLAYING GAME)*.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2010). *Computational Thinking: What and Why?*

Zahid, M. Z. (2020). Telaah kerangka kerja PISA 2021: era integrasi computational thinking dalam bidang matematika. *PRISMA, Prosiding Seminar Nasional Matematika*, *3*, 706–713. https://journal.unnes.ac.id/sju/index.php/prisma/