Script Sensei

A vector-based algorithm to correct Japanese kanji

Ellis Dijkstra

UNIVERSITY OF TWENTE.

Supervisor	Marcus Gerhold
Critical Observer	Gwenn Englebienne
Study Program	Creative Technology (BSc)
University	University of Twente
Country	The Netherlands
Date	2024

Abstract

Learning to write Japanese kanji comes with the implicit challenges of continuous practice of writing kanji and the necessity of revisiting learned kanji. For this purpose, Script Sensei has been developed – a vector-based algorithm designed to check and correct written kanji. This project assessed the possibility of a vector-based algorithm to achieve this objective. This was investigated in multiple steps: creating a website to gather input strokes from the user; reducing the input strokes to turning points and inflexion points; converting the turning points and inflexion points into vectors; comparing the vectors with the templates from the *KanjiVG* database [1]; and finally, displaying the feedback based on count, order, direction, shape and size. Suggestions included improving the accuracy of the simplification and improving the method to display the feedback; however, Script Sensei is a suitable alternative aimed at accurately learning to write kanji. With its innovative approach, Script Sensei showcases the potential of vector-based algorithms to enhance the learning experience for learners seeking to master the substantial challenge of writing Japanese kanji.

Acknowledgement

I would like to thank my Critical Observer, Gwenn Englebienne, for supporting me throughout this project.

I would like to extend special thanks to my Supervisor, Marcus Gerhold, for his dedication in guiding me throughout this project. During our weekly meetings, he gave constructive criticism, made lighthearted jokes, and restored my confidence when necessary.

Additionally, I would like to thank Ysbrand Burgstede for functioning as a rubber duck during debugging sessions.

Contents

CHAPTER 1.

Introduction

Japanese foreign language learners encounter difficulties when presented with the Japanese character-based writing system, especially if their native language employs an alphabetic script [2]. The writing system is so difficult to grasp, that it even deters people from learning the language to higher levels of proficiency [3]. The predicament of learning the writing system lies in its usage of three different scripts: *kana*, *kanji* and *romaji* (see Figure 1.1). Romaji is the Roman alphabetisation of the Japanese language, which was created to improve education and literacy and to create a way to interact with computers. Kana consists of two different scripts: *hiragana* and *katakana*. Both scripts consist of 46 characters and represent the same set of sounds. The difference lies in their usage: hiragana is used for native Japanese words and katakana is used for words originating from other languages. Kanji, however, represent a whole word or whole idea and consist of over 40.000 characters, out of which a person needs to know at least 2.136 kanji for official everyday use [2]–[4]. The difficulty with learning Japanese lies in mastering the vast number of kanji.

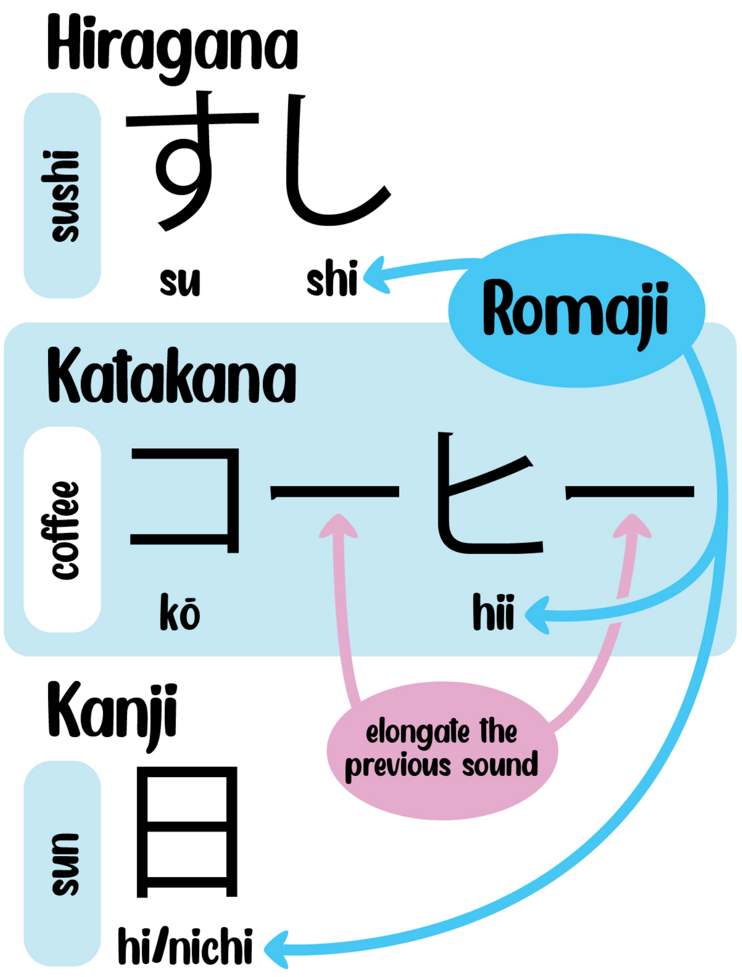


Figure 1.1: Examples of the four Japanese scripts.

When teaching Japanese as a Foreign Language (JFL), students show a disconnection in the acquisition of their reading and writing skills and their overall proficiency [3]. Although students might know how a word sounds and be able to write it in kana, Japanese texts use kanji. Authentic reading materials, even when aimed at children, have a wider range of used kanji than a student would be familiar with. This makes the reading of authentic materials either difficult or impossible. This problem is equivalent to when students want to write a text in Japanese, as they cannot write a character that they do not know. Even if they knew what the character looked like, the student is faced with additional difficulties, as Japanese characters are written with prescribed stroke order and stroke direction [3], [5]. This makes the writing of Japanese characters exceptionally difficult.

The most popular teaching method – both for JFL students and native Japanese children – is rote learning [6], [7], a memorization technique based on repetition. Due to the large number of kanji that are written, checking them manually becomes increasingly time-consuming. Additionally, the mistakes are often hard to spot. Although it is hard to point out if a student followed the right stroke order and stroke direction when writing, not adhering to these conventions makes the kanji unbalanced and sometimes even difficult to read. How well the character is written shows the proficiency of the writer and therefore affects the respect that the reader has for the writer [8]. Automating the correction of handwritten Japanese characters could provide a solution for the aforementioned problems. Thus, this paper examines possible options *to create an application that is capable of effectively educating users on the correctness of written Japanese characters*.

1.1 Research Questions

Main Research Question:

> What enhancements can be applied to the current software that can effectively educate users on the correctness of written Japanese characters?
> How can an application be created that can effectively educate users on the correctness of written Japanese characters?

Sub Research Questions:

1. What are the best practices and common pitfalls when learning Japanese?
2. What are the best practices and common pitfalls in the design of educational software for language learning?
3. What are potential approaches for developing algorithms capable of recognising Japanese characters on a stroke-based level?
4. What are the features and limitations of applications available in the Netherlands to teach users how to write Japanese characters?

CHAPTER 2.

Background Research

Based on the challenges outlined in the introduction, the following section dives deeper into the necessary background research for each of the previously mentioned subquestions. First, the difficulties of learning how to write Japanese characters are inventoried, hereafter the helpfulness of educational software is explored, after which possible algorithms are analysed, and related work is considered.

2.1 Learning the Japanese writing system

The difficulty of learning Japanese lies in mastering the vast number of kanji. The learner must learn 2.136 kanji to be functionally literate. Kanji, a *morphographic script*, represent the smallest unit of meaning in a word. This is a script that most of the world is not familiar with, as most languages use alphabetic or syllabic scripts [2]. Since each word has its unique character, Everson [9] shows that it is hard to decipher unknown kanji; although the kanji might have some components which the reader recognises, the meaning and pronunciation remain difficult to perceive. Additionally, Rose [2] establishes that most kanji can be read or pronounced in multiple ways, depending on the context and their use. Another complication of learning kanji is that there are six different types (see Table 2.1). Applying just one memorisation technique for all types of kanji will lead to far-fetched examples to make the technique work [2]. This means, that besides the extensive number of kanji one must learn, it is difficult to learn the kanji too.

Kanji type	Usage	Example kanji	Meaning	Effective learning strategies
Pictographs	Pictorial representations	木	A stylized *tree*	Pictorial Association
Logograms	Abstract representations	赤	The color *red*	
Ideographs	Combinations of pictographic components	休	A combination of the kanji for person and tree, symbolizing *rest*	Component Analysis combined with other strategies
Semasio-phonetic ideographs	Combinations of components	明	A combination of the kanji for day and night, symbolizing *tomorrow*	Component Analysis combined with other strategies
Derivative characters	Derived from disassociated concepts	楽	This kanji means *comfort*	
Phonetic loan characters	Adopted for phonetic reasons	来	Originally meant as a pictograph for wheat, it now means *to come*	

Table 2.1: The six types of kanji with examples and effective learning strategies.

Rose [2], [3] establishes that it is best to use (a combination of) learning strategies depending on the kanji type the student wants to learn (see Table 2.1). Researchers agree that the effectiveness of memorisation techniques depends on the type of kanji being learned. Bourke has found that there are 15 different categories of kanji learning strategies (as cited in [2], [3]). Rose [2], Kandrac [10] and Chikamatsu [11] state novice Japanese learners mostly encounter *pictographs* – kanji that are pictorial representations of objects – for which *visual association* strategies are best suited, such as *pictorial association* and *symbolic association*. Pictorial association ties the shape of the kanji to images (see Figure 2.1) and symbolic association relates the kanji to the Roman alphabet, Arabic numerals, or a character from one of the kana. *Component Analysis* can be used to break the two types of *ideographs* into components; the smallest parts a kanji can be split up into are *graphemes*, which can be memorised using another strategy. A good way to study ideographs, for example, is by coming up with a *mnemonic*; a story that gives that kanji a meaningful interpretation. The kanji for "rest" can be memorised with the mnemonic "a person under a tree is resting", tying the components "person" and "tree" to the kanji "rest". How effective these strategies are depends on the specific kanji– not every type of kanji has a specific strategy that works perfectly. The remaining types of kanji are the *logograms, derivative characters*, and *phonetic loan characters*. These characters do not stand for concrete ideas, making them harder to learn. Logograms, for example, are symbolic representations of abstract ideas. Here, visual associations are less useful, as trying to produce a drawing to symbolise the kanji for "red" will prove difficult. The actual learning method for these types of kanji depends on the student and their preferred way of learning. Since every person learns differently, not every strategy is universally effective. Therefore, Rose [2] suggests learners educate themselves about learning strategies before starting to study kanji, so students have a bigger repertoire of strategies available to them.

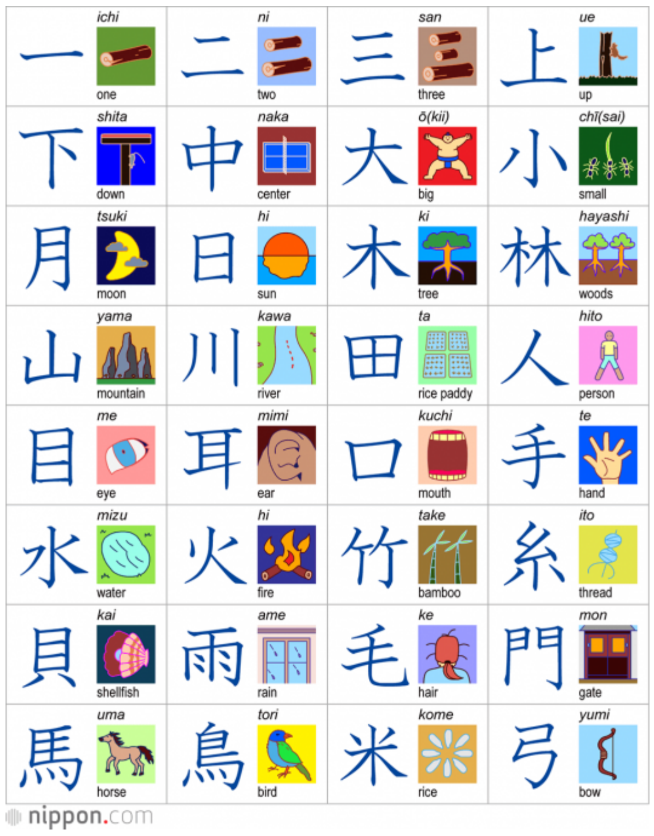


Figure 2.1: Examples of pictographic associations for kanji. [12]

Although researchers agree that using multiple learning strategies benefits the learners, they are divided on prioritising spoken or written language acquisition. Rose [2] notes that it is important that students actively try to memorise the material. This causes a deeper encoding of the knowledge in the student's memory, which makes it more likely to be remembered. Another way a deeper encoding can be achieved is by making meaningful connections with existing knowledge. The positive effect of meaningful connections is supported by Matlin and Roediger *et al.*, who point out that information can be recalled more accurately when related to other associations, images, or experiences (as cited by Rose [2]). Rose mentions that this deeper encoding can be accomplished by prioritising spoken language acquisition over written language acquisition. Rose adds that students will be familiar with the meaning of the kanji when learning the character – and therefore able to make a more meaningful connection – by prioritising speaking. However, prioritising spoken language acquisition is not a viable option. Rose [3] points out in other work that students develop their knowledge of spoken Japanese faster than their written knowledge and notes that this results in a problem, exhibited by students knowing a word, but not being able to write it. Rose claims this disparity is caused as authentic Japanese materials, even when aimed at children, are written with a wider range of kanji than the student knows, making the materials inaccessible. Although this is disproven by Hitosugi and Day [13] – who executed a research experiment for which the 14 participating students read a minimum of 16 authentic children's books in ten weeks -, the difference in spoken and written language skills is confirmed by Chikamatsu [11]. Thus, giving priority to spoken language acquisition is not a viable option for students, as it will make the disparity between written language acquisition and the student's overall proficiency level bigger, which might lead to problems in the long run.

The struggle of mastering kanji does not stop at simply studying the kanji. To write a kanji correctly on paper, one must be able to recall the character with sufficient accuracy. Native Japanese speakers find more and more often that this process has become slower and more difficult, even for kanji they once mastered during their school years. The cause of this lies in the modernisation of technology, as Japanese writers interact with devices by using romaji and get possible kanji suggested. As the writers only have to recognise the character to be able to use it, they encounter problems when they have to write the kanji themselves. The phenomenon of being unable to produce a kanji due to it being on the fringe of recollection has been explored by Chikamatsu [11] and named *'tip-of-the-pen'*, after the similar occurrence of 'tip of the tongue'. This phenomenon seems unique for languages that use a morphographic script and might even occur mid-kanji. As Rose and Harbon [14] found in the case of second language learners, this phenomenon is likely to exhibit itself when students are trying to become advanced learners of the Japanese language. This also means that being functionally literate in Japanese requires continuous dedicated retrieval practice to combat forgetting the learned kanji. Advanced learners could be deterred from learning Japanese as the number of kanji that needs to be revised increases [2].

2.2 Creating educational software

Although practising writing kanji on paper feels the most natural, it has its challenges. When an instructor is presented with a written kanji, it is hard to evaluate whether the stroke order and direction are correct. This could be solved by monitoring the students while writing the kanji, though this is tedious for the teacher. Another solution is requiring the students to enumerate the beginning of the strokes to check the order and direction, which is unnatural and increases the students' workload [8], [15]. Consequently, Taele and Hammond [15] argue that automating the assessment of kanji and providing feedback on the structure and technique benefits students. This can be achieved by the creation of software to evaluate Japanese characters.

To make educational software more entertaining, gamified elements can be incorporated. Compared to other software, Hamari *et al.* [16] found that gamification was most often used for educational software. The effect of gamification was mostly positive, as it could lead to increased motivation, engagement, and enjoyment of learning. However, negative outcomes – such as increased competitiveness – were also noticeable. This indicates that gamification is a valid option to enhance the application, but should be implemented conservatively. Morschheuser *et al.* [17] found seven steps to the gamification of projects. These steps are: forming clear objectives regarding problems and goals, analysing the context and user, ideating, designing prototypes, implementing a design, evaluating, and monitoring the approach. In addition to these steps, Morschheuser *et al.* listed several essential requirements for successful gamification projects. They repeat the necessity of analysing the context and user, creating clear objectives, and monitoring after the release. Additionally, they stress early testing of gamification approaches. Furthermore, they recommend considering legal and ethical constraints during the design phase and eliminating possible options for cheating. Lastly, they point out that gamification is more complex than adding points, achievements, and leaderboards, but do not elaborate further.

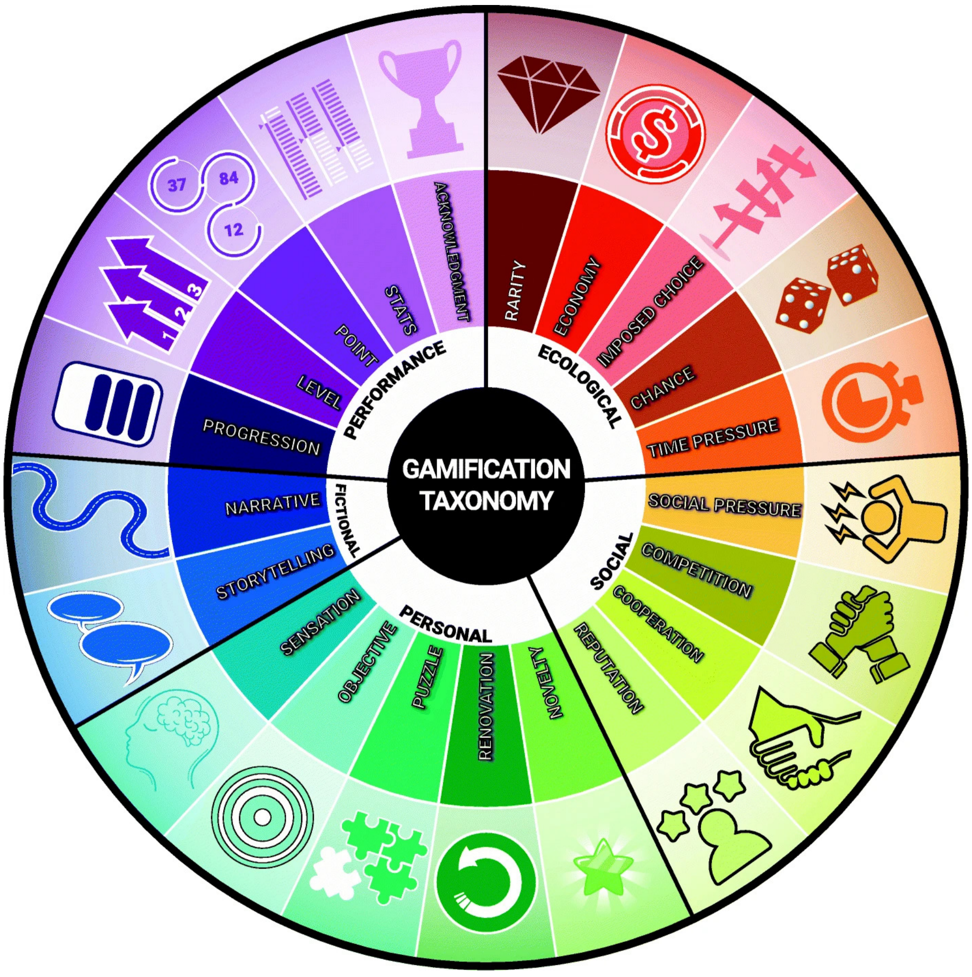


Figure 2.2: A taxonomy of all possible gamification elements [18].

However, extensive research has been executed by Toda *et al.* [18] regarding different gamification elements. All elements and their dimensions can be seen in Figure 2.2. The value of the dimensions and the elements and their explanations can be found in the list below.

- **Performance/measurement**
    - **If included:** Provides feedback to the learner.
    - **If excluded:** Students may feel disoriented.
    - **Elements:**
        - *Acknowledgements* praise specific actions of players and can be badges, medals, trophies, and achievements.
        - *Levels* in skill and character provide new advantages and challenges.
        - *Progression* allows users to determine their progress.
        - *Points* are the simplest method of gamification
        - *Stats* (statistics) are scores based on the number of tasks completed or on the acquired skill of players.
- **Ecological**
    - **If included:** Allows users to interact with elements.
    - **If excluded:** Makes the environment feel dull.
    - **Elements:**
        - *Chance* can be used to assign points or bonuses randomly.
        - *Imposed choices* allow users to choose how they advance in the environment.
        - *Economy* allows players to make transactions.
        - *Rarity* stimulates users to acquire limited resources.
        - *Time pressure* is used to pressure the learners' actions, but can potentially disengage learners.
- **Social**
    - **If included:** Allows students to interact with each other.
    - **If excluded:** Students might feel isolated.
    - **Elements:**
        - *Competition* allows users to best other users.
        - *Cooperation* requires users to collaborate on achieving a common goal.
        - *Reputation* allows users to accumulate titles, which can create a hierarchy.
        - *Social pressure* places pressure on the learner due to social interactions.
- **Personal**
    - **If included:** Provides meaning for the learners.
    - **If excluded:** Can demotivate the user.
    - **Elements:**
        - *Novelty* avoids stagnation to prevent disengagement and demotivation.
        - *Objectives* provide players with a purpose to perform the required tasks.
        - *Puzzles* are related to the activities implemented in the environment.
        - *Renovation* allows users a second chance at a failed task and is one of the properties that makes games fun.
        - *Sensations* are stimulations of vision or sound.
- **Fictional**
    - **If included:** Ties the users' experience to the context.
    - **If excluded:** Causes loss of meaning or context.
    - **Elements:**
        - *Narrative* dictates the order of the elements in the game.
        - *Storytelling* is how the story's background is told.

Toda *et al.* analysed the importance of each of these elements regarding educational applications. In the measurement dimension, they found that all elements are important: without *acknowledgement*, the user feels as if their actions are unimportant; *levels* are used to show *progression*, which the user needs to see their advancement and to prevent feelings of frustration and anxiety; and *stats* help the user to feel oriented. However, Toda *et al.* add that achievements must be used correctly to prevent unexpected behaviour and that there is no consensus yet on using the elements correctly.

The ecological dimension should be implemented carefully to prevent deterring the user rather than motivating them. It is suggested to implement *chance* so that the user always succeeds after a certain number of tries. The *economy* should be related to the content and could lead to advantages in lessons. A careful balance needs to be found between too many and too few *imposed choices* or items with increased *rarity*. Lastly, the absence of *time pressure* can lead to boredom. A suggestion for time pressure is to provide flexible deadlines.

The social dimension asks for careful design too, as this dimension influences the dynamics between players. *Competition* can either be very effective or demotivating depending on the learner's performance. Good practices for competition are creating groups or not tying it to content-based activity. *Cooperation* is a positive influence, although it is harder to design. Toda *et al.* add that cooperation may result in the sharing of knowledge between students or an increase in motivation to avoid putting their peers in jeopardy. A lack of *reputation* may lead to the feeling that actions are not meaningful. *Social pressure* is seen as most irrelevant, but can be helpful if properly designed.

The personal dimension is very prevalent in educational software due to the nature of the application. *Objectives, puzzles*, and *renovation* are always present. Objectives display themselves as the objective is to learn something; puzzles are all cognitive tasks, often implemented as tests; and (almost) all educational applications allow users to revisit certain tasks or knowledge. Objectives are the most relevant element to use within gamified educational applications. *Sensation* is qualified as highly relevant and can be achieved by creating a pleasant interface. *Novelty* is useful but can be quite complex to integrate due to the need to regenerate test questions.

Lastly, the fiction dimension is not often considered for educational software. However, creating a *narrative* and using *storytelling* to shape this narrative, may help the user focus on the application's content.

2.3 Algorithms for Japanese character recognition

Although general considerations for educational software have now been found, the application will need to be able to recognise Japanese characters as well. This can be achieved by writing an algorithm for Japanese character recognition. A possible approach for recognising kanji is to evaluate the kanji's contour rather than its strokes. One of such algorithms is written by Hartono and Ginting [19]. This algorithm uses Linear Spatial Filtering to reduce the pixels of the kanji to its contour. This contour is then converted to a chain of pixels, after which the location (left, right, up, down, or one of the four diagonal options) of the next pixel is stored and counted. This counted number is then converted to a percentage and the Manhattan distance is used to find the most similar character. The *recognition rate* of an algorithm is the percentage of characters the algorithm can recognise. For this algorithm, the recognition rate is low (60.26%) and the algorithm has only been tested on five hiragana characters, not to mention that the algorithm struggles when the kanji is written in a different shape, size, or position. Analysing contours proves ineffective and does not allow feedback on the stroke order and direction of the written kanji.

Instead, it is easier to analyse the stroke by reducing it to a thin line – also known as *skeletonisation* – and simplifying it as much as possible. Taele and Hammond [15] have followed this approach and assume all kanji can be simplified until they can be approximated with lines. They use an algorithm from [20] to find the turning points of individual strokes; allowing the strokes to be analysed as lines. This approach is comparable to the approach from An and Li, who have added a visualisation in their report (see Figure 2.3). Additionally, Taele and Hammond [15] describe the kanji's shape with components and constraints. The components are related to the physical shape of the stroke, whereas the constraints are the orientation, the order of the points, and the spatial relationship to other strokes. The constraints are defined by language constructs from LADDER (as cited in [15]). In this paper, Taele and Hammond do not explain how these components and constraints can be used to compare the input to templates, stating the recognition rate to be 92.9% and their algorithm to give feedback on the stroke order and direction flawlessly. Taele and Hammond [22] improved upon their algorithm a year later. The approach remained similar, but a third algorithm was added – the PaleoSketch algorithm (as cited in [22]). This algorithm can, among others, find the inflexion point of an arc, allowing for better comparison of complicated shapes. Again, the algorithm's specifics are not discussed, but presumably, the authors compared the components and constraints with those of the template characters to find the closest match. This algorithm had a recognition rate of 95% and still corrected without mistakes. Thus, stroke recognition works better when the minimalisation of the strokes does not cause the inflexion points of the strokes to be disregarded.

Alternatively, strokes can be classified and compared with the template. An approach from Kobayashi *et al.* [23] uses skeletonisation (using the method of Deutsch, as cited in [23]) to find the middle of the strokes. These are then analysed to find the endpoints, the junctions where strokes overlap, and the inflexion points. The segments are the lines between any of these points. The segments are always straight; any arcs in the character are ignored. Afterwards, strokes are created from the segments, until each segment is connected to two endpoints. Kobayashi *et al.* normalise the strokes to make their orientation either horizontal, vertical or diagonal and their position is slightly adjusted. Each stroke is then analysed to find its central point, length (short, medium, or long), and orientation (horizontal, vertical, or either of the diagonals). These points are then compared to the templates. Kobayashi *et al.* report a recognition rate of 92.1%, but they do not check the technique of the written kanji. It is unclear how this would work for characters with curved strokes.

Most algorithms compare characters based on the physical properties of the strokes, making it harder to recognise strokes that are incorrectly placed or sized – mistakes often made by novice learners. Hence, Chu *et al.* [24] used Dynamic Time Warping (DTW), an algorithm capable of comparing sequences with similar patterns but differing lengths and stroke placements. Another difference in approach appears when the stroke count between the input and template does not align. In such a case, Chu *et al.* determined four types of errors which a user can make in a stroke: missing, adding, concatenating and splitting. Consequently, a incorrect stroke count does not necessarily mean that strokes have been added or forgotten – thus these strokes must be checked too based on the shape, size, and direction. Chu *et al.* have found a solution for this. When comparing the characters, all matching strokes from the input and template are added to a list; then a greedy algorithm is used to add the rest of the strokes based on the smallest Hausdorff distance[1]. If the stroke counts of the characters do not match, a one-to-many correspondence is sought. This algorithm has a recognition rate of 86%.

An and Li [21] have a similar approach as Chu *et al.*, although An and Li store the characters as strokes of at least one segment. A stroke consists of more than one segment if there are turning points (see Figure 2.3). An and Li match strokes based on the distance between the centres of mass, the length of the stroke and the aspect ratios of the bounding box. Unmatched strokes are split into segments and then compared with the unmatched segments of the template. If the segments are similarly positioned, they are merged to check if they match the template. The

[1]The Hausdorff distance is the maximum distance between the closest points from two sets of points

algorithm of [21] has a recognition rate of at least 87%, depending on the number of strokes the character has. Despite the lower recognition rates compared to previously mentioned algorithms, the approaches from Chu *et al.* and An and Li seem better fit to recognize characters written by novice learner, as the algorithm compares the way the character is written rather than the physical properties of the shape.

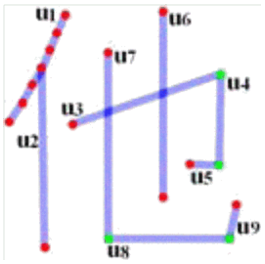


Figure 2.3: The endpoints of the strokes (red) and the turning points of the strokes (green) [21].

Although it is most common to compare the strokes themselves with the strokes of the template, it is not the only possibility. A completely different approach was taken by Nguyen *et al.* [8], who started by extracting the important points from strokes as done by Ramer (as cited in [8]) to represent strokes with a minimal number of points. The input strokes are compared to the template's to reduce the number of points further. The strokes are then filled with evenly spaced points. The character is then placed on top of a grid, after which the number of points in each grid area is calculated. The resulting number is compared with the numbers acquired for the template characters. The recognition rate of this algorithm is 98.5%. This approach leads to the highest recognition rate of Japanese characters and therefore seems the best approach for character recognition.

2.4 Related work

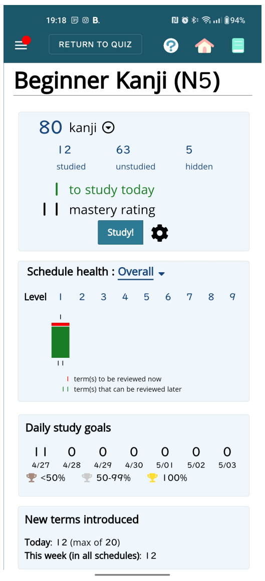
Although the research about JFL learners is limited, there are ample applications available that can teach a user Japanese. Even though not all of the software is written with a focus on writing, there are quite some ingenious approaches the developers came up with. Ochi *et al.* [25], for example, have written an application that analyses a text with kanji. It displays the kanji the user should be familiar with but provides the hiragana for the kanji the learner has not learned yet. Lin *et al.* [26] have made a handheld device that can record and playback mnemonics for specific kanji so users can stick to their preferred learning strategies. Especially being able to record your own mnemonics might be useful for a user's kanji study.

Different papers do have a focus on writing, all with distinct approaches. Berque and Chiba [27] wrote software for classical teaching, allowing the teacher to write a kanji on their tablet, which is simultaneously displayed on the students' tablets. Furthermore, teachers can play back the writing of the students' characters to check how students wrote the characters. Lastly, the software allows the teacher to discuss the characters with the class. This is less fitting for students who are learning individually. Bhattacharya and Bhattacharya [28] are creating a tool to teach students both the meaning of the kanji and how to write the kanji, but want to accomplish this by making only use of pictorial strategies. At the time of writing, they admit they are still working on this, as they have trouble figuring out how the more complicated kanji can be derived towards pictures. Previous research discourages using a single strategy to learn kanji, making the application unfit.

There are, however, papers about applications that help acquire writing skills individually. Fathoni and Delima [29] gamified the learning of kanji writing. However, users can only practise writing with a template, which does not prevent the tip-of-the-pen phenomenon. It is unclear if the app gives instructions and/or corrections to write the kanji, as it is only mentioned that the app teaches kanji to the user. The app from Ng *et al.* [30] teaches users how to pronounce and write a character. It generates questions rather than using preset exercises and corrects users on their handwriting. They also implemented an adaptive learning system that individualises the learning process for individuals. Similar to the app made by Fathoni and Delima, a user is shown a template to write the characters. Additionally, all possible answers are generated similarly, making it easy for the user to find the right answer when this is discovered. Lastly, the app only focuses on hiragana, disregarding katakana and kanji. Another example is the software from Taele, Koh *et al.* [31]. Their tool teaches users extensively what a Japanese character means and checks the written character on several points, but – just like the past two examples – supplies a template to trace. The application from Taele and Hammond [15] works similarly to the rest. In addition, it allows users to practise writing the kanji that are part of composed kanji individually, allowing additional practice for the difficult parts of the kanji. However, it also shows the kanji the user needs to write. Ishida and Shin [5] made software that teaches users about the kanji in detail as well. They also admit that students do not stop to look at the provided feedback, leaving room for improvement. The last reviewed paper is written by Iwayama *et al.* [32], who created software that checks the writing of kanji but does not give immediate feedback, making it harder to improve upon made mistakes. In conclusion, the related work often shows the kanji to be drawn, which does not prevent the tip-of-the-pen phenomenon.

There are also applications available on the Google Play Store that could offer suitable alternatives. One of the most well-known examples of such an application is Duolingo, which allows you to learn many languages in one application. Duolingo focuses on learning whole languages rather than aspects of it. Additionally, "Duolingo [...] is one of the most successful examples of gamification in education" [18, p. 8]. However, when writing Japanese characters in Duolingo, the user gets the starting point of the first stroke supplied, which hints towards the character's shape. Furthermore, the written strokes are corrected immediately, and sometimes even incorrect strokes are accepted as correct. This does not allow the user to try to write the character as they think it should be, and it does not allow for corrections after writing the kanji if the user realises it is incorrect. Conclusively, Duolingo does not prevent the tip-of-the-pen phenomenon.

Although multiple applications are available on Google Play that facilitate Japanese learning, not all of them include the teaching of writing skills. Presumably, this is due to the difficulty of implementing algorithms to correct Japanese characters. These applications are "HeyJapan: Learn Basic Japanese", "N5-N1 JLPT test – Migii JLPT", and "Bunpo: Learn Japanese". Other applications use templates, which makes correcting the kanji easier. "Write It! Japanese" uses a template, but only developed their application for the kana. "JA Sensei – Leer Japans JLPT" wrote an application that gives a lot of details about the learned kanji, such as the meaning, the meaning of the graphemes, and other kanji the kanji is used in. However, they also only practise on top of a template.



Beginner Kanji (N5)

80 kanji

12 studied 63 unstudied 5 hidden

1 to study today

11 mastery rating

Schedule health : Overall

1 term(s) to be reviewed now

11 term(s) that can be reviewed later

Daily study goals

11	0	0	0	0	0	0
4/27	4/28	4/29	4/30	5/01	5/02	5/03

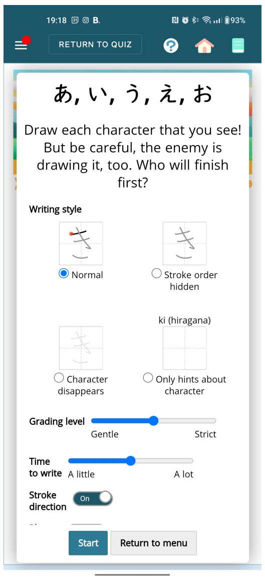
<50% 50-99% 100%

New terms introduced

Today: 12 (max of 20)

This week (in all schedules): 12

(a) The statistics.



あ, い, う, え, お

Draw each character that you see! But be careful, the enemy is drawing it, too. Who will finish first?

Writing style: [✓ Normal] Stroke order hidden; Character disappears; Only hints about character

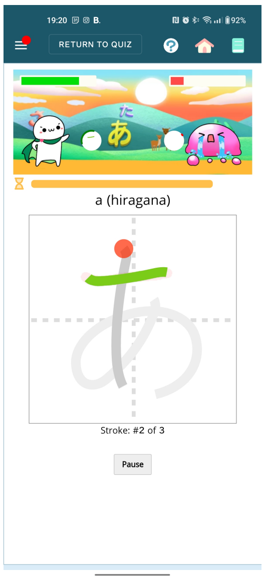
ki (hiragana)

Grading level: Gentle — Strict

Time to write: A little — A lot

Stroke direction: On

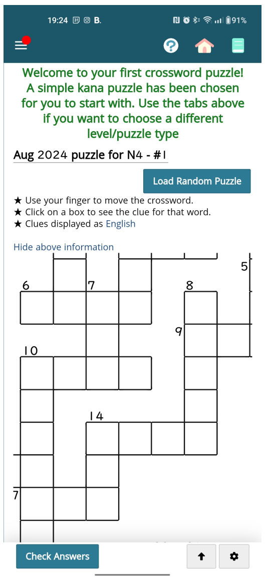
(b) Settings for the writing game.



a (hiragana)

Stroke: #2 of 3

(c) The writing game with half-answered kanji.



Welcome to your first crossword puzzle! A simple kana puzzle has been chosen for you to start with. Use the tabs above if you want to choose a different level/puzzle type

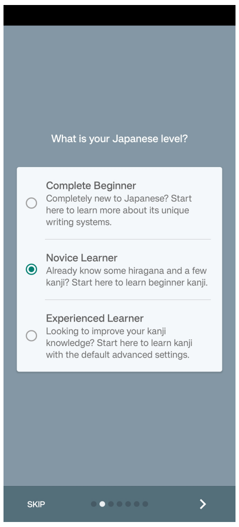
Aug 2024 puzzle for N4 - #1

★ Use your finger to move the crossword.
★ Click on a box to see the clue for that word.
★ Clues displayed as English

(d) The crossword.

Figure 2.4: The design of "Renshuu".

There are also a few applications whose executions are incredibly similar to the desired result of this project. "Renshuu" created several games to test the user on their Japanese (see Figure 2.4). The application is not developed to teach the user Japanese but does quiz the user in a fun way with games, such as speed writing, quick counting, and crossword puzzles. For the writing, the user can set the difficulty of writing the kanji. However, the application does not allow the user to write the whole character but rather corrects the user after every stroke the user writes. This is a similar approach as taken by "Japanese kanji Study" (see Figure 2.5): they do not check the whole character, but rather correct the user per stroke. Additionally, not all the content of the application is freely available.

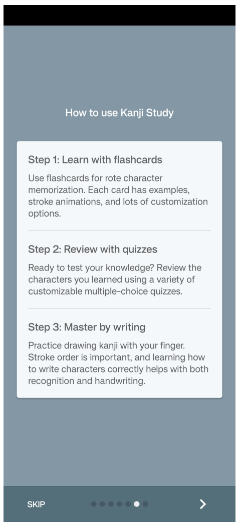


What is your Japanese level?

- Complete Beginner
  Completely new to Japanese? Start here to learn more about its unique writing systems.
- [✓ Novice Learner]
  Already know some hiragana and a few kanji? Start here to learn beginner kanji.
- Experienced Learner
  Looking to improve your kanji knowledge? Start here to learn kanji with the default advanced settings.

SKIP

(a) Personalisation of the app.



How to use Kanji Study

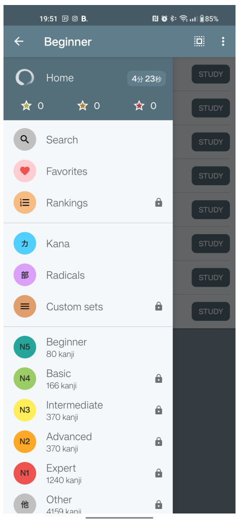
Step 1: Learn with flashcards
Use flashcards for rote character memorization. Each card has examples, stroke animations, and lots of customization options.

Step 2: Review with quizzes
Ready to test your knowledge? Review the characters you learned using a variety of customizable multiple-choice quizzes.

Step 3: Master by writing
Practice drawing kanji with your finger. Stroke order is important, and learning how to write characters correctly helps with both recognition and handwriting.

SKIP

(b) Explanation of the teaching method.



19:51 85%

Beginner

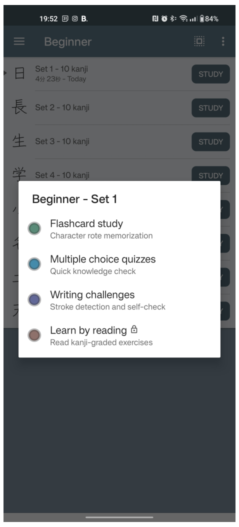
Home 4分 23秒

☆ 0 ☆ 0 ☆ 0

Search
Favorites
Rankings
Kana
Radicals
Custom sets

N5	Beginner	80 kanji
N4	Basic	166 kanji
N3	Intermediate	370 kanji
N2	Advanced	370 kanji
N1	Expert	1240 kanji
他	Other	4159 kanji

(c) Possible learning options.



19:52 84%

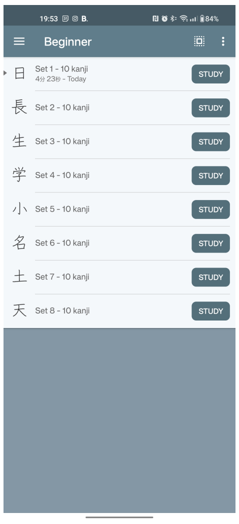
Beginner

日 Set 1 - 10 kanji 4分 23秒 - Today STUDY
長 Set 2 - 10 kanji STUDY
生 Set 3 - 10 kanji STUDY
学 Set 4 - 10 kanji STUDY

Beginner - Set 1

- Flashcard study
  Character rote memorization
- Multiple choice quizzes
  Quick knowledge check
- Writing challenges
  Stroke detection and self-check
- Learn by reading
  Read kanji-graded exercises

(d) Quizzing options.

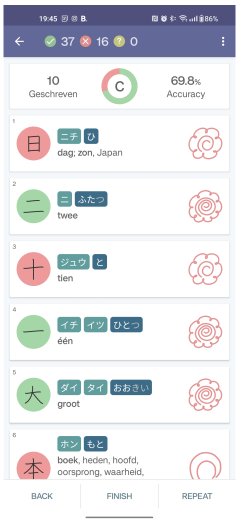


19:53 84%

Beginner

日	Set 1 - 10 kanji 4分 23秒 - Today	STUDY
長	Set 2 - 10 kanji	STUDY
生	Set 3 - 10 kanji	STUDY
学	Set 4 - 10 kanji	STUDY
小	Set 5 - 10 kanji	STUDY
名	Set 6 - 10 kanji	STUDY
土	Set 7 - 10 kanji	STUDY
天	Set 8 - 10 kanji	STUDY

(e) Sets of kanji.



19:45 86%

✓ 37 ✗ 16 ? 0

10 Geschreven C 69.8% Accuracy

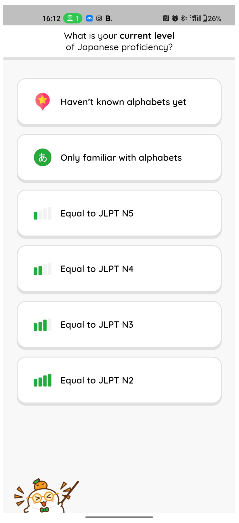
1	日	ニチ ひ	dag; zon, Japan
2	二	ニ ふたつ	twee
3	十	ジュウ と	tien
4	一	イチ イツ ひとつ	één
5	大	ダイ タイ おおきい	groot
6	本	ホン もと	boek, heden, hoofd, oorsprong, waarheid,

BACK FINISH REPEAT

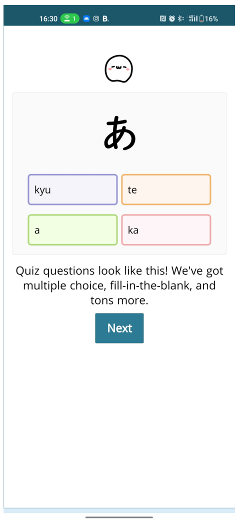
(f) Score after making a test.

Figure 2.5: The design of "Japanese Kanji Study".

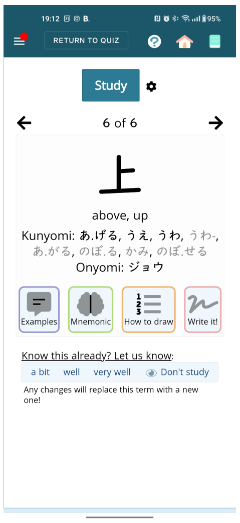
The last reviewed application is "Mochikanji" (see Figure 2.6), which allows users to write whole kanji and have them checked afterwards. The only minor point of disturbance was that Mochikanji does not show where the mistake in the character is located until it just shows the user the whole character. It also seemed that Mochikanji allows users to pass the quizzes too easily. After retaking the exam, it became clear that only the final character is checked. The stroke order and direction are not corrected, and characters consisting of four strokes written with a single stroke were also deemed correct. This means that the user does not learn how to write the kanji correctly, they learn which shape they should be writing. Additionally, the length of the animations is rather long, which takes a lot of time.


What is your current level of Japanese proficiency?
Haven't known alphabets yet
Only familiar with alphabets
Equal to JLPT N5
Equal to JLPT N4
Equal to JLPT N3
Equal to JLPT N2

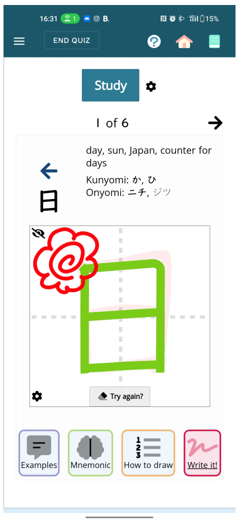
(a) Personalisation of the app.


あ
kyu
te
a
ka
Quiz questions look like this! We've got multiple choice, fill-in-the-blank, and tons more.
Next

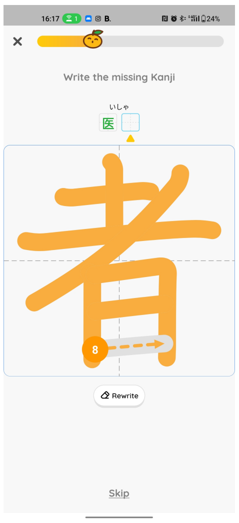
(b) Example of knowledge questions.


RETURN TO QUIZ
Study
6 of 6
上
above, up
Kunyomi: あ.げる, うえ, うわ, うわ-, あ.がる, のぼ.る, かみ, のぼ.せる
Onyomi: ジョウ
Examples
Mnemonic
How to draw
Write it!
Know this already? Let us know:
a bit
well
very well
Don't study
Any changes will replace this term with a new one!

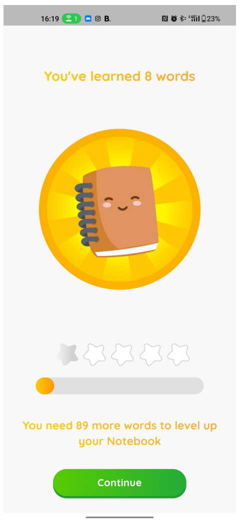
(c) How kanji are taught.



(d) How writing the kanji is taught.



(e) A test question for writing kanji.



(f) Gamifying elements to motivate the learning of kanji.

Figure 2.6: The design of "Mochikanji".

2.5 Summary

The related work could answer each of the subquestions for this project. The following section repeats each research question and gives the provided answer.

SQ 1. What are the best practices and common pitfalls when learning Japanese?

The first subquestion of this research showed that learning to write in Japanese comes with quite a few challenges; the number of kanji that one needs to know is vast, and learning kanji is optimal when multiple learning strategies are applied. Additionally, the lack of accessible authentic reading materials makes it harder for learners to progress equally in acquiring spoken and written language skills. The last difficulty lies in the repetition of the ever-growing number of kanji the students need to reiterate to combat forgetting them.

SQ 2. What are the best practices and common pitfalls in the design of educational software for language learning?

To create an appealing application to educate users on the correct writing of Japanese characters, at least one element out of each dimension (as described by Toda *et al.* [18]) should be picked to prevent encountering the problems outlined in section 2.2. Obvious choices for educational software are objectives, puzzles, and progression.

SQ 3. What are potential approaches for developing algorithms capable of recognising Japanese characters on a stroke-based level?

The analysis of this subquestion showed that multiple algorithms are available to support novice language learners. The best approaches simplify strokes to possible turning points and inflexion points in addition to their start- and endpoints, after which either the strokes are compared individually or based on their location on a grid. It depends on the researcher's intention if they decide to take an approach based on DTW, such as the algorithm by Chu *et al.* [24], which is the best fit for novice learners, or an approach based on the grid as used by Nguyen *et al.* [8], which has the highest recognition rate.

SQ 4. What are the features and limitations of applications available in the Netherlands to teach users how to write Japanese characters?

Although there are several applications available to learn kanji, each researched application has its benefits and shortcomings. Ranging from the showing of a template to the incomplete correction of kanji, these imperfections give way to creating an application which does not have these limitations. However, these applications serve well as inspiration for this project.

CHAPTER 3.

Ideation

The ideation phase helps to evolve the design question into a product idea. This can be achieved through stakeholder requests or based on existing technology and creative ideas. Since existing technologies have been outlined in section 2.4, ideas sparked by these technologies will be treated as creative ideas. Stakeholders are individuals, groups, or organisations who are interested or invested in the process and/or outcome of a project. Analysing stakeholders is essential in determining the requirements and desired result of the project. Creative ideas help make a product meaningful and innovative. Multiple ideation techniques, such as brainstorming, sketching, and prototyping, can provoke creative ideas. Other methods include making storyboards, mock-ups, or word webs. For this project, the preferred strategies are making word webs and mock-ups, and sketches will be used to explore visual ideas. These strategies can help form a coherent product idea with possible development objectives.

3.1 Stakeholder analysis

The primary group of stakeholders are the users of the application. These stakeholders are interested in the application because they will use it to practise writing kanji. However, they have little influence on the design and functionality of the application. Although possible suggestions can be provided during feedback sessions or user testing, these suggestions are not guaranteed to be implemented in the eventual application. The users can be split into three groups depending on how they use the application. Most users are expected to be JFL students, who will use the application to practise writing kanji repetitively. Another group includes JFL teachers, who might use the application during lessons or as a supporting tool for homework. Lastly, native Japanese speakers could use the application to review learned kanji. Repeating kanji combats the tip-of-the-pen phenomenon [11], as forgotten kanji will be easier to recollect after repetition. For all groups of users, the application's usability and capability to check and correct written Japanese characters are important.

3.2 The algorithm

Earlier in this chapter, several possible ideation methods are named. However, the idea for this algorithm developed naturally during chapter 2. In this section, the evolution of this idea will be recounted as accurately as possible.

As noted in section 2.4, a common element is found in most researched applications. Twelve applications focus on writing, of which six use a template for the user to trace. Additionally, Duolingo guides users to the starting points of their strokes, and – just like two other applications – corrects written strokes before the whole character is completed. The reason most of these applications use hints can be explained from a technical standpoint. Supplying a template guides users to begin strokes at the right position, to write all strokes in the correct size, and to follow the right shape. Giving the starting point for the first stroke means that the stroke can be checked relative to this initial point instead of starting with an empty canvas. Similarly, correcting individual strokes immediately after writing them, allows algorithms to check the newest stroke relative to the previous stroke. This makes the algorithms of these applications less complicated. However, to optimise the learning of kanji, this algorithm should allow users to write the entire character completely from memory to prevent the tip-of-the-pen phenomenon.

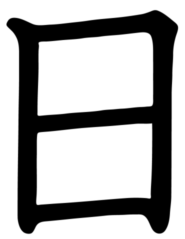
There are multiple ways to write an algorithm which meets these criteria. Inspiration for the algorithm stems from the different approaches in section 2.3. Although the approach for this algorithm can be similar, the final goal differs. The studied algorithms mostly aim to recognise kanji; this algorithm knows which character should be written and needs to correct it instead. This also means that many of the researched algorithms can assume that the kanji are written (mostly) correctly; this algorithm must be able to handle both correct or incorrect kanji. Checking the character becomes more difficult when the user has made a mistake in writing the kanji, regardless of whether the user incorrectly wrote a different stroke, grapheme, or kanji. Additional errors could be made in the position, size, orientation, or shape of a stroke. Finally, users can make mistakes by adding, removing, concatenating, or splitting strokes. Due to the possibility of this wide range of errors in the physical properties of the strokes, an algorithm written to compare the sequences and patterns of characters will probably be a better fit for the target audience of this application.

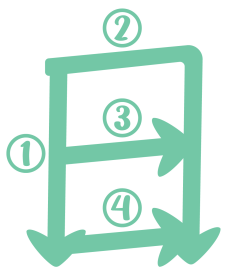
As mentioned earlier, the approach for the algorithm has been figured out naturally during chapter 2. The first problem an algorithm encounters when analysing a stroke is that strokes often have a width. Therefore, skeletonisation can be used to reduce the stroke to its thinnest version. This skeleton is then analysed. This approach is used often in chapter 2 and seems a good starting point. After skeletonisation, the algorithm needs to either compare the physical properties of the stroke – which is often used for character recognition – or compare the patterns – which is more useful for novice learners [24]. A pattern recognition algorithm is a better fit. One of the characteristics of these patterns is that stroke position and length should not cause trouble for the algorithm. The algorithms from An and Li [21], Kobayashi *et al.* [23] and Chu *et al.* [24] mentioned the use of vectors to check the direction of a stroke, but no algorithms solely used vectors to compare the template character to the input character. Additionally, no results could be found when searching for such algorithms. Consequentially, using vectors solely to compare the template with the input strokes originated.

This idea was elaborated to see if problems could be foreseen, as it presents both advantages and difficulties to overcome. The benefits of using vectors are the reduced relevance of direction, position, and size of the kanji's elements. The direction can be checked by seeing if the angles of the vectors differ by less than π. If this difference is bigger, all vectors can be inverted and the stroke can be analysed from the last point instead of at the beginning. Secondly, the position of elements becomes less important, as vectors are based on a directional element rather than their position. The position can still be checked by comparing the centres of masses of the elements or by following the approach from An and Li [21]. Lastly, the size can be checked by comparing the length of individual vectors. Moreover, checking if strokes are split or concatenated becomes easier. After checking all strokes against each other, the remaining strokes of both characters can be analysed per vector to see if additional matches can be found.

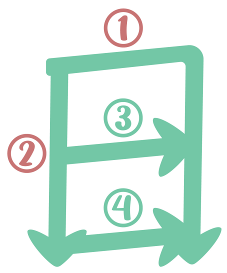
A disadvantage is that the points of which the shape consists should be picked carefully; if the input character and the template character are drawn similarly, but the input character picks another point as inflexion point, the vectors will not match either. This problem is similar to simplifying a curve to four points when it consists of only three in the template. Another issue regarding the shape arises from small mistakes in the writing which split a single stroke into two vectors. The first two problems should be solvable with the algorithm from Sezgin *et al.* [20], or the Paleosketch algorithm (as cited in [22]). A solution for the second problem might present itself later in the process.

The algorithm will need to be able to recognise different mistakes in the written kanji. Different examples of common mistakes can be seen in Figure 3.1. Figure 3.1a shows how the character for "Sun" looks if it is written correctly; Figure 3.1b shows the correct stroke order and direction. Figure 3.1c is written with a incorrect stroke order; Figure 3.1d is written with stroke 3 in the other direction. In Figure 3.1e stroke 3 has been written incorrectly. In Figure 3.1f stroke 1 is a concatenation of strokes 1 and 2 in Figure 3.1b. Stroke 2 has been split in Figure 3.1g, but since strokes 2 and 3 were written in succession, the stroke order is still deemed to be correct. Figure 3.1h shows a character where the same stroke is split, but the stroke order is incorrect. The feedback for these characters can be given in different ways: the direction is either correct or incorrect; the size, shape, and location can be graded with a score; and the count and the order of the strokes can be shown by the offset with count and the correct order of the strokes respectively.

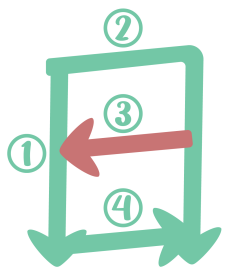

(a) The character for "Sun".



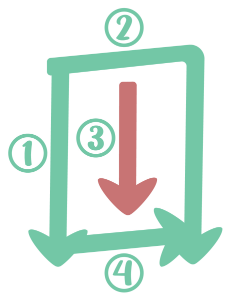
(b) The correctly written character for "Sun".



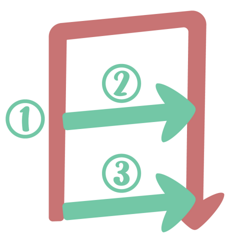
(c) The character "Sun" with a wrong stroke order.



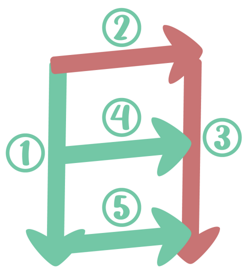
(d) The character "Sun" with the third stroke in the wrong direction.



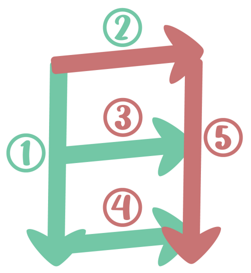
(e) The character "Sun" with the third stroke written wrong.



(f) The character "Sun" with the first and second stroke merged.



(g) The character "Sun" with the second stroke split into two.



(h) The character "Sun" with a split stroke and a wrong stroke order.

Figure 3.1: Possible ways of writing the kanji for "Sun".

3.3 The application

The user still needs a way to interact with the algorithm, which can be achieved by creating an application. Although this project focuses on the algorithm, ideation for an application proved easier. Thus, this section will describe ideas that could be implemented if time is left. The preferred strategy for ideating the application was the creation of mind maps. Multiple aspects had to be considered, including the application's features. The possible features could be roughly divided into four categories: usability, classwork, learning, and hints (see Figure 3.2). Although there will not be enough time to implement all these features, considering them helps shape a possible application and prioritise which features should be implemented.

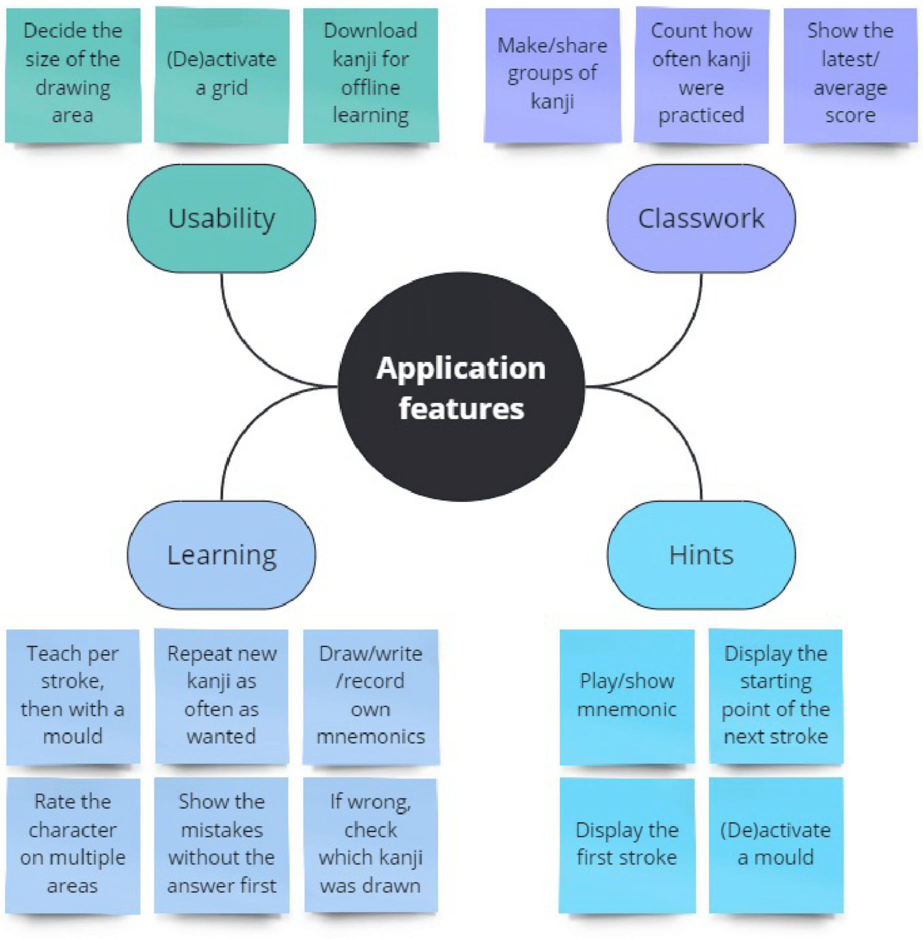

Decide the size of the drawing area
(De)activate a grid
Download kanji for offline learning
Make/share groups of kanji
Count how often kanji were practiced
Show the latest/ average score
Usability
Classwork
Application features
Learning
Hints
Teach per stroke, then with a mould
Repeat new kanji as often as wanted
Draw/write /record own mnemonics
Play/show mnemonic
Display the starting point of the next stroke
Rate the character on multiple areas
Show the mistakes without the answer first
If wrong, check which kanji was drawn
Display the first stroke
(De)activate a mould

Figure 3.2: Feature brainstorm for this project.

The category "usability" consists of three possible features. The first two have to do with the accessibility of the writing area – the space in which the user writes the kanji. The first is to allow users to determine the size of the writing area. Depending on the used input device, the user could benefit from having a small or a large writing area. It also allows the user to pick a size of writing they would like to become familiar with, as the user could have a small or big handwriting. Lastly, a larger writing space may be preferred, if the user uses a sizeable stylus or a finger. The second feature, overlaying a grid over the writing area, can also benefit the user, as it could make placing the kanji in the writing space easier. The third feature allows the user to download specific kanji to learn offline.

The second category, "learning", consists of features that should help the user to master the kanji. Firstly, a good approach from section 2.4 to teach the kanji, was to start by adding individual strokes to trace, then by showcasing a template to trace, and then with an empty canvas. It would benefit the user to repeat writing the kanji as often as they want. Another feature stemming from section 2.4 is to let the user draw, write, or record their mnemonics, to make learning the kanji more individualised. Regarding the correction of the kanji, multiple features could be implemented to benefit the user. The first is to show how well the user did by rating the character in several areas, such as the number of strokes, the position of strokes, the shape of strokes, the size of strokes, the number of hints needed, and the overall correctness of the kanji. If the kanji was written incorrectly, the user could get an optional second chance to write the correct kanji after showing the user their mistakes. That way, if users are in doubt between two kanji, they can still try to write the correct kanji the second time, or they can retry writing the parts labelled as incorrect. Lastly, if another kanji was written, displaying the meaning of the written kanji might help to reinforce the differences between the two kanji.

If the user still has trouble figuring out which kanji they should be writing, there are multiple hints they could use in chronological order, as shown in the category "hints". For example, the user could play or show the mnemonic they drew, wrote, or recorded. If this is not enough of a hint, the user could have the option to display the starting point of the next stroke. After this, the whole stroke could be shown, or the user could activate a template to trace the character. However, if the template is used, the kanji should be marked as (mostly) incorrect so the user can practise it again.

The least interesting features to implement in this project are the features regarding "classwork". However, if time is left at the end of the project, these features could increase the application's usability in classical settings immensely. For example, users could make and share the groups of kanji they are learning with their class. Additionally, depending on the homework the teacher gave the students, the app could display how often each kanji was practised. Something that would help the student figure out which kanji need more attention is displaying the latest and/or average score achieved for each kanji.

3.4 Gamification

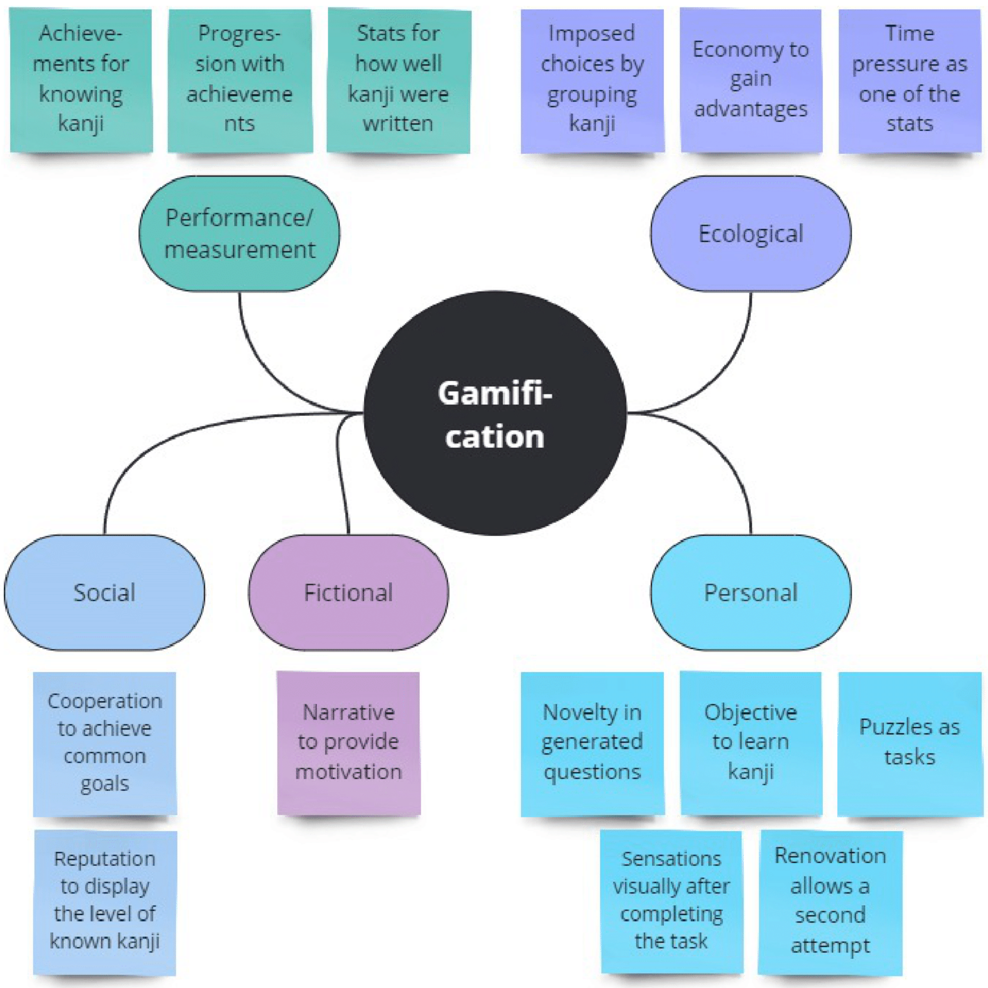

Achieve-ments for knowing kanji
Progres-sion with achieveme nts
Stats for how well kanji were written
Imposed choices by grouping kanji
Economy to gain advantages
Time pressure as one of the stats
Performance/ measurement
Ecological
Gamifi-cation
Social
Fictional
Personal
Cooperation to achieve common goals
Narrative to provide motivation
Novelty in generated questions
Objective to learn kanji
Puzzles as tasks
Reputation to display the level of known kanji
Sensations visually after completing the task
Renovation allows a second attempt

Figure 3.3: Brainstormed features for this project.

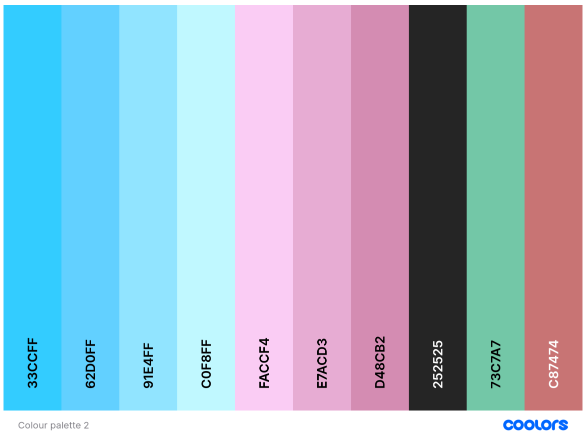
Based on the literature research on gamification, a second mind map was made based on the elements and categories as constructed by Toda *et al.* [18] (see Figure 3.3). From the performance category, achievements, progression, and stats were picked. Achievements could motivate users to practise for a certain amount of time, a certain number of times, or to know a certain number of kanji. The latter could also be a way to show the user their progression. Progression could also be shown with a progress bar or by making a virtual landscape through which the user advances as they learn more kanji. Lastly, stats could be used to show the user how well they wrote the kanji.

One of the chosen elements related to the ecological category is imposed choice. This can be implemented by allowing users to choose which kanji they want to be learning next. Allowing users to pick the kanji, allows the users to study the kanji they are treating in their lessons. Time pressure is said to either persuade users to complete the lessons or to demotivate them [18]. As such, time could be used as a criticism to grade the performance on the kanji. This would mean that the user is inclined to draw the kanji quickly if they want to achieve a high score, but their progress is not dismissed if they are too slow. The least important element to implement is economy. It could motivate the user to gain higher scores on the learned kanji, but this can be achieved otherwise as well, for example by showing the latest scores of the kanji on the overview of learned kanji.

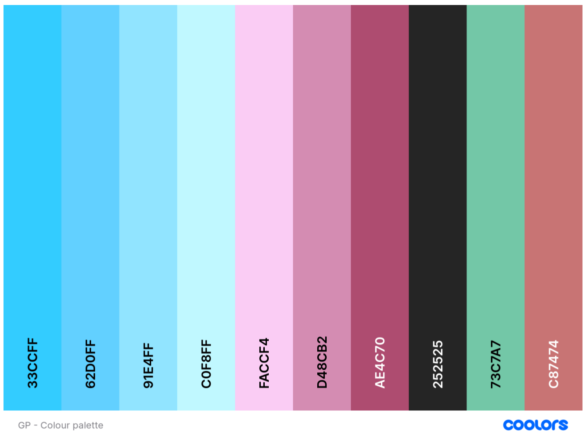
As with all learning applications, objective and puzzles are naturally included due to intrinsic motivation and the usage of tests and quizzes [18]. Sensations could be added by displaying an encouraging message after completing a task. As described above, renovation could be used to allow users a second attempt after writing a kanji incorrectly the first time. Novelty was suggested to be included by generating questions to answer rather than reusing the same questions repeatedly [18]. However, this might be out of scope for an application that focuses solely on the writing of kanji.

Social elements could be implemented to improve classical teaching, for example by using cooperation to help classmates achieve common goals. Individually, reputation could be used to be able to show other users quickly how much experience one has with kanji. Narrative is a fictional element that could be implemented to create a story for users to either follow or unlock the next parts.

3.5 Design



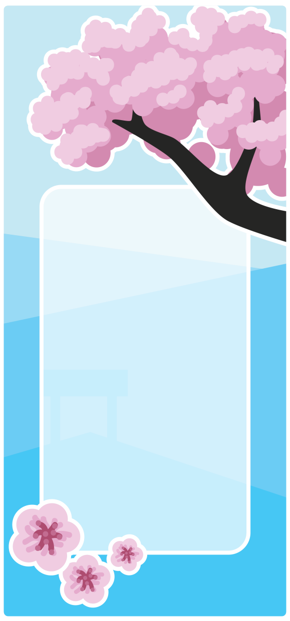
(a) The light colour palette.



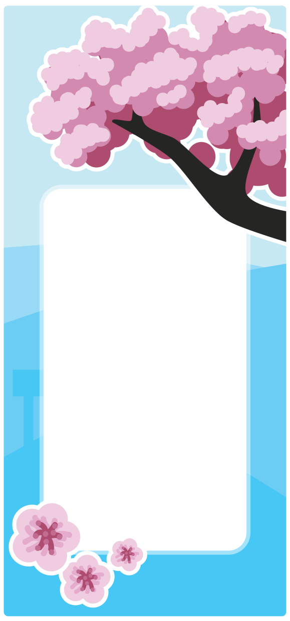
(b) The dark colour palette.

Figure 3.4: The options for the colour palettes for the applications design.

For the design of the application, a colour palette was created (see Figure 3.4). The pink colour is inspired by the blossoms of the Sakura tree. Either blue or a green tint could have been picked as a complementing colour; blue to make the background feel like the sky. Additionally, green can be implemented with red to show if characters were written correctly or not. A slight variation in the pink tones was made, to experiment with the amount of contrast preferred for the application.



(a) The application's design with the light colour palette (see Figure 3.4a) and a transparent background.



(b) The application's design with the dark colour palette (see Figure 3.4b) and a white background.

Figure 3.5: The first designs for the application.

The usage of the colour palettes and the first design of the application can be seen in Figure 3.5. Things to consider are the large decorations, which take up a lot of space, and therefore do not allow for a lot of text to fit on the screen. There are two variations visible in Figure 3.5. Figure 3.5a shows the use of the lighter colour palette in the Sakura tree, and uses a transparent textbox. The transparent textbox was picked to show the background of the design better, although it might be too distracting for the users. Therefore, in Figure 3.5b, the textbox is white and the outline of the textbox is transparent. Additionally, the dark colour palette was used to draw the Sakura tree.

CHAPTER 4.

Specification

The specification aims to develop the current idea into an executable concept. This is accomplished by enhancing chapter 3 to create an executable project idea. Although the emphasis of this project lies in the creation of the algorithm, the user interaction is easiest when supplied with an application. Although the original idea during chapter 3 was to use a mobile application, a website was created instead. In the interest of usability, a mobile application would be more user-friendly, as it is easier to write accurate strokes on a mobile application than on a web application - especially if the user does not possess additional input devices. Additionally, if the user were to purchase an input device to operate this application, a mobile stylus would be cheaper than a drawing tablet. However, creating a web application decreases the development time, due to the higher proficiency of the researcher. Therefore, a website was created using *Flask* [33]. *Flask* allows a Python program to set up an HTML environment on the computer's localhost. This HTML environment can then be used to program the website, while the algorithm itself can be written in Python.

4.1 Requirements

An important part of the specification process is the requirements set for the algorithm's behaviour. Requirements can be split into Functional Requirements (FRs) and Non-Functional Requirements (NFRs). Functional Requirements describe what the system should do and Non-Functional Requirements describe how the system should do it. Both types of requirements are ordered due to their priority for the algorithm.

4.1.1 Functional Requirements

FRs describe the behaviours and functions a system must perform. They are often similar to the needs and expectations that the end-users of the product experience. As FRs often specify the inputs and outputs of a system, including the processes and overall system behaviour, the necessary FRs can be found by the inexplicit requirements determined in chapter 3 and by creating a time-sequence diagram and analysing the user's interaction with the system (see Figure 4.1). For example, FR 5 and FR 8 stem from the time-sequence diagram and the other requirements are based on chapter 3.

Functional Requirements:

☐ FR 1. The algorithm *must* check written Japanese characters.
☐ FR 2. The algorithm *must* correct written Japanese characters.
☐ FR 3. The algorithm *must* not give any hints regarding how the kanji *should* be written.
☐ FR 4. The algorithm *must* allow the user to write the whole kanji before checking it.
☐ FR 5. The algorithm *must* allow users to select which kanji they want to practise.
☐ FR 6. The algorithm *must* be able to check the kanji based on stroke order and direction.
☐ FR 7. The algorithm *must* check the kanji independent of the correctness of the size and the positioning of the kanji.
☐ FR 8. The algorithm *must* be able to recognise mouse input.
☐ FR 9. The algorithm *should* be able to check the kanji, even if the kanji is partially correct.
☐ FR 10. The algorithm *should* be able to recognise strokes consisting of both straight lines and curves.
☐ FR 11. The algorithm *should* be able to recognise input from a drawing tablet.
☐ FR 12. The algorithm *could* provide feedback by giving a rating on multiple areas (count, order, direction, shape, size, and location).
☐ FR 13. The algorithm *could* integrate with external databases for additional information about the kanji.
☐ FR 14. The algorithm *could* include a feature to save and review past practice sessions.
☐ FR 15. The algorithm *could* support multiple users with individual progress tracking.

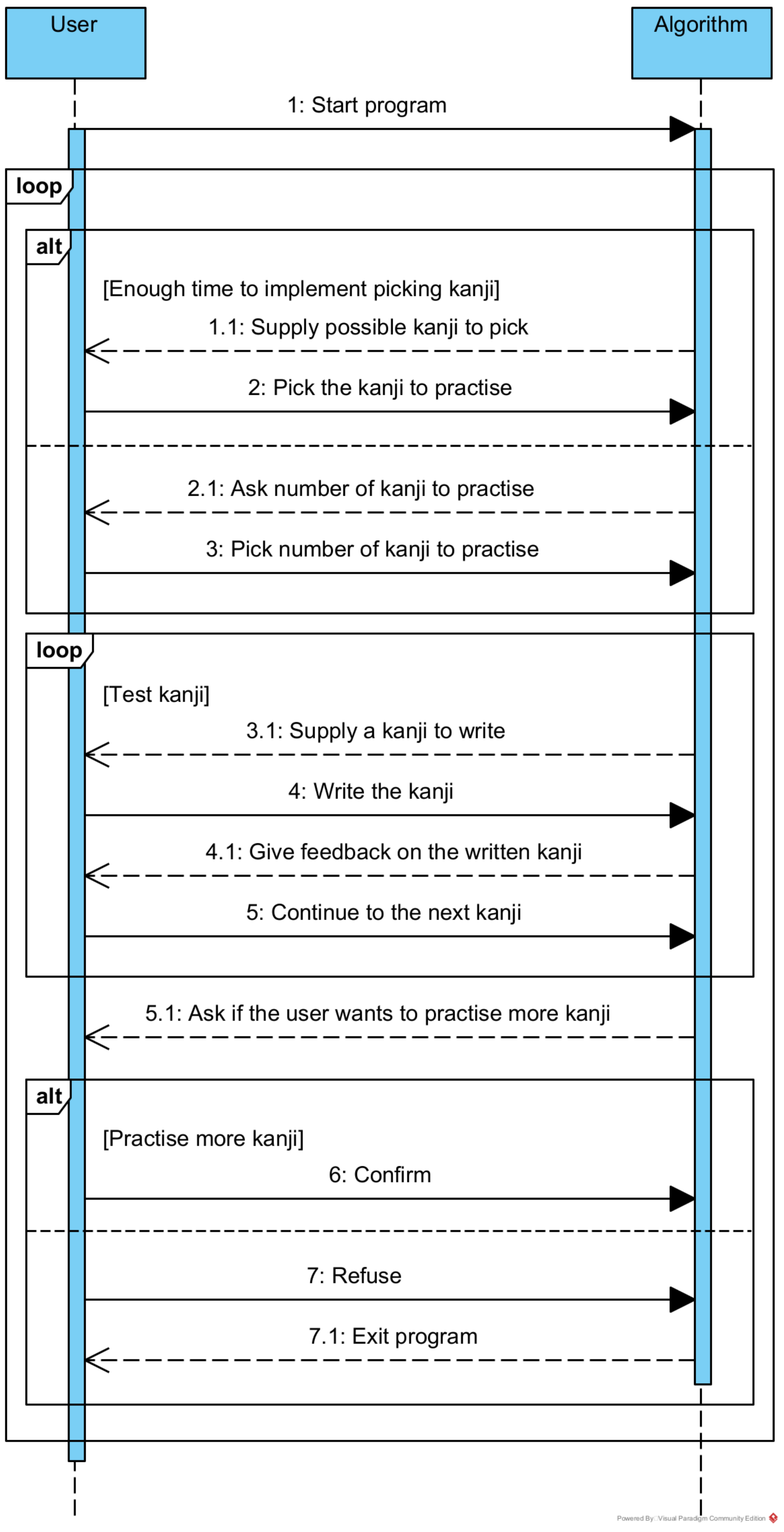

User
Algorithm
1: Start program
loop
alt
[Enough time to implement picking kanji]
1.1: Supply possible kanji to pick
2: Pick the kanji to practise
2.1: Ask number of kanji to practise
3: Pick number of kanji to practise
loop
[Test kanji]
3.1: Supply a kanji to write
4: Write the kanji
4.1: Give feedback on the written kanji
5: Continue to the next kanji
5.1: Ask if the user wants to practise more kanji
alt
[Practise more kanji]
6: Confirm
7: Refuse
7.1: Exit program
Powered By: Visual Paradigm Community Edition

Figure 4.1: The sequence diagram that shows the interaction with the algorithm.

4.1.2 Non-Functional Requirements

While FRs focus on what the system does, NFRs define how the system performs operations to ensure that the system stays resilient, efficient, and user-friendly. Aspects that should be considered when outlining the NFRs include performance, usability, reliability, and security [34]. Requirements regarding usability are NFR 2, NFR 4, and NFR 6; NFR 3 is a metric for the performance and NFR 5 stems from reliability. Security is not applicable for this project, as the website does not store data for multiple sessions. NFR 1 is based on scalability, another aspect which can be considered for websites.

Non-Functional Requirements:

☐ NFR 1. The algorithm *must* be able to expand the number of kanji it supports.
☐ NFR 2. The user interface *should* be easy to navigate.
☐ NFR 3. The algorithm *should* respond to the user in less than five seconds.
☐ NFR 4. The algorithm *should* be compatible with web browsers.
☐ NFR 5. The algorithm *should* give similar feedback to similarly written kanji.
☐ NFR 6. The user interface *could* ensure usability for users with disabilities.

CHAPTER 5.

Realisation

5.1 Databases

The basis of this algorithm revolves around a database of kanji. This database should include at least the 2.136 kanji used daily, as it would form the basis of the visual appearance of the stroke. It should also provide information about the stroke order and direction of individual strokes so the algorithm can check this as well. Although databases with handwritten kanji are rather easy to find, they are inappropriate for this purpose. Instead, a digital database should be found that represents kanji in individual strokes, with the correct shape, direction, and stroke order. *KanjiVG* [1] is a database that fulfils these requirements. The database consists of 11.658 SVG files containing information about the kanji. This number includes slight variations in writing style for specific kanji, allowing the user to pick which style of kanji is preferred. The kanji are based on Japanese schoolbook fonts, meaning that the SVG files most likely align with the way the kanji are taught in classes. The SVG files contain information about the points in the kanji, the stroke order, the stroke direction, and more. Although this additional information is unnecessary for this project, these three attributes are extremely useful. Therefore, *KanjiVG* was selected as the database for this project.

5.2 The website

The website consists of several pages, designed in the same style. It uses the colour palettes shown in Figure 3.4, as the cherry blossoms looked better with additional details that could not be expressed with three colours. The style of the header is consistent throughout the pages, just as the font and the general design. The first page the user sees is the selection page, after which they continue to the overview and writing pages.

5.2.1 The selection page

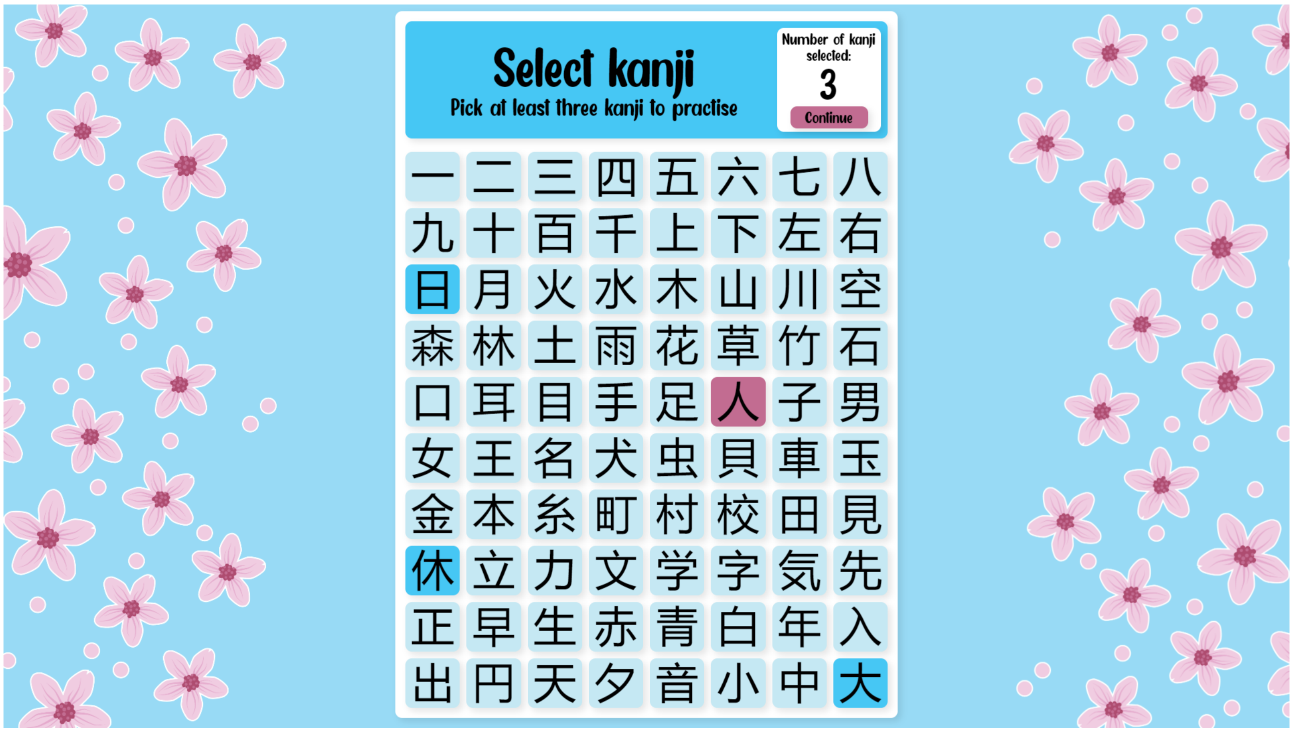


Figure 5.1: The select page of the website, with three selected kanji and the cursor hovering over a kanji.

Although it would be possible to provide the user with several random kanji to learn, the user benefits most by being able to select which kanji they would like to practise writing. Therefore, the selection page was added (see Figure 5.1). This page shows the user the available kanji which they can practise. As this algorithm is a prototype, a subset of 80 kanji was added rather than the entire database. These 80 kanji were selected, being the kanji taught in the first year of elementary school in Japan [2]. This list, however, seemed to have no apparent order; therefore, it was split into several categories: numbers, directions, natural elements, body parts, people and related concepts, animals, objects, locations, and abstract concepts (see Appendix B). The user is asked to pick at least three kanji to practise writing with, after which they move to the overview page.

5.2.2 The overview page

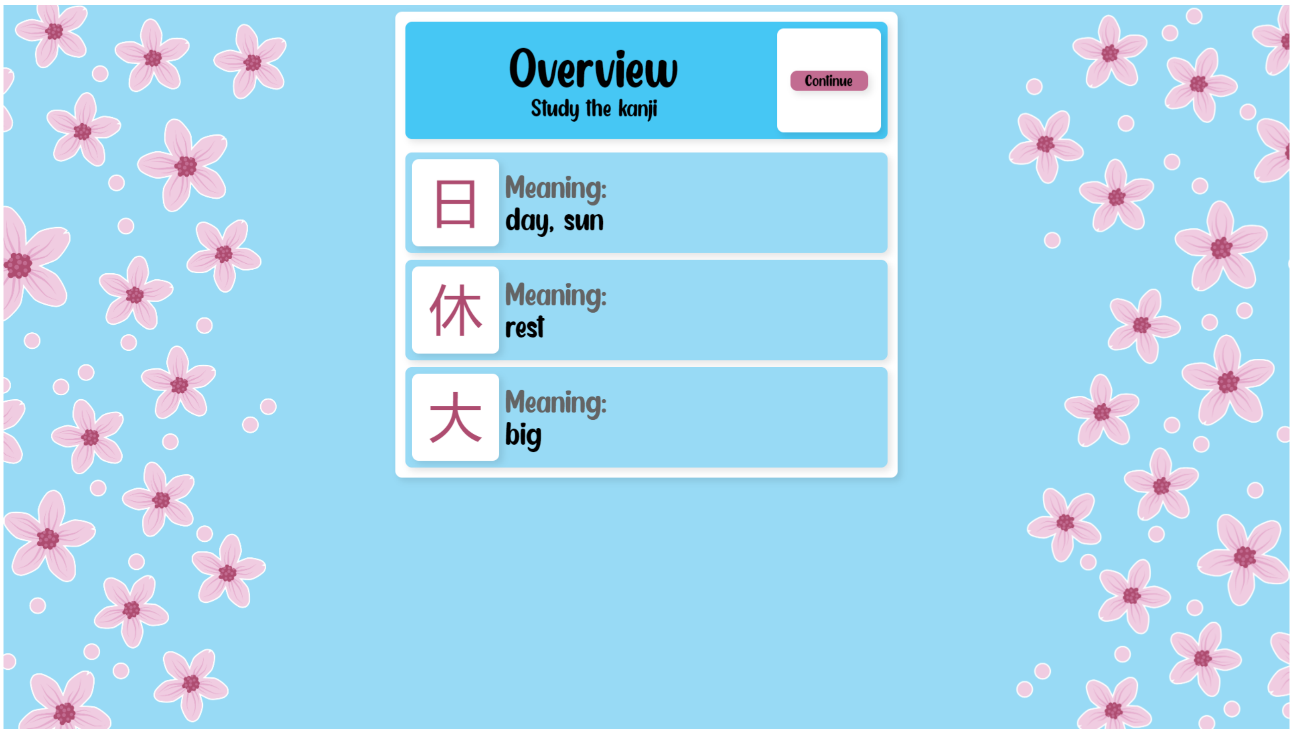


Figure 5.2: The overview page of the website, with the meaning of the three selected kanji from Figure 5.1 displayed.

The overview page allows users to familiarize themselves with the meaning of the selected kanji (see Figure 5.2). The meaning of each kanji was acquired from the website *Kanji alive* [35]. The overview page shows the kanji without information about the stroke order and direction. It allows for the physical appearance of the kanji to be recalled, making the first kanji on the writing page easiest to answer. Thus, a minimum number of kanji is set on the selection page. Showing a single kanji on the overview page to be immediately written on the next page, does not prevent the tip-of-the-pen phenomenon. The choice to have a minimum of three was a difficult one. Preferably, the number should be as high as possible, as the time between recollecting and drawing the kanji will become higher for each kanji, preventing the tip-of-the-pen phenomenon. However, picking a high number will decrease the usability for novice learners, as they have not mastered large numbers of kanji yet. Additionally, picking a high minimum number will cause practice sessions to increase in duration. Of course, there are still ways to work around this minimum number of kanji. Technically, the user could pick two mastered kanji or two simple kanji to meet the requirement of selecting three kanji. This would allow them to focus on the single remaining kanji and therefore not avoid the tip-of-the-pen phenomenon. However, as the user will be using the website due to intrinsic motivation, the user is expected to pick two other difficult kanji instead, which can then be practised on the writing page.

5.2.3 The writing page

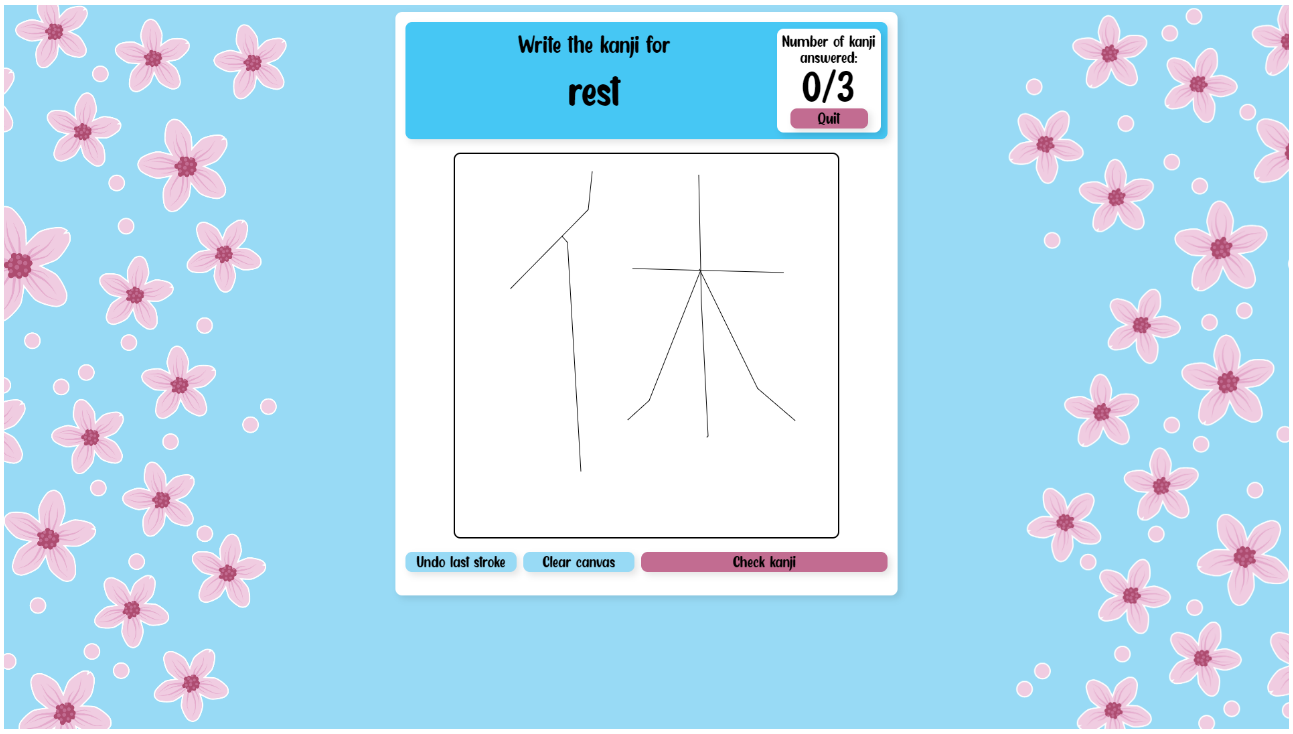
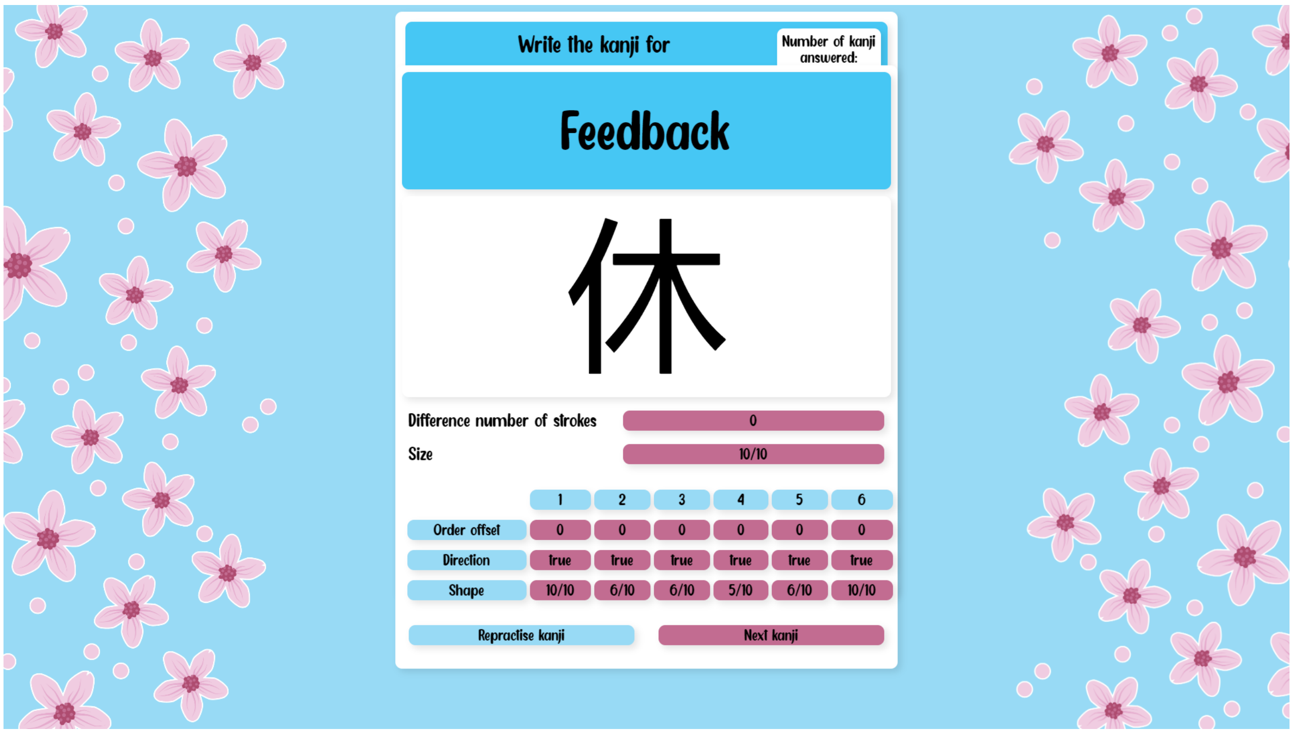


Figure 5.3: The writing page of the website, with the kanji for rest (休) written.

This page allows users to write the kanji (see Figure 5.3). The page displays an SVG canvas which records the user's strokes. An input device aimed at writing, such as a drawing tablet, increases the ease of use. The meaning of the kanji is displayed at the top of the page, so the user knows which kanji they have to write. The kanji itself is not visible on the page. At the bottom of the screen, the user has three buttons: undo the last stroke, clear the canvas, or check the kanji. The first two help the user with the writing of the kanji; the latter opens a pop-up that shows the feedback the algorithm gives to the user (see Figure 5.4). This pop-up shows what the desired kanji looks like, and provides scores regarding the number of strokes, the proportions of the elements, the stroke order, the stroke direction, and the accuracy of the shape. At the bottom, the user gets two choices: rewrite the current kanji, or move on to the next.


Write the kanji for

Number of kanji answered:

Feedback

休

Difference number of strokes	0
Size	10/10

	1	2	3	4	5	6
Order offset	0	0	0	0	0	0
Direction	true	true	true	true	true	true
Shape	10/10	6/10	6/10	5/10	6/10	10/10

Figure 5.4: The feedback page of the website, displaying the feedback for the kanji written in Figure 5.3. During user testing, "Difference number of strokes" was called "Stroke offset".

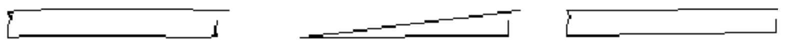
5.3 The algorithm

The algorithm consists of two different parts: stroke simplification and stroke comparison. Due to the use of *Flask*, various programming languages were used throughout the project. The main page of the website which redirects the HTML links is written in Python, just as the algorithm which compares the kanji. The webpages are written in HTML, CSS and JavaScript. The algorithm for simplification is written in JavaScript. Lastly, the paths are displayed using an SVG canvas and SVG paths, of which the latter have a specific structure to adhere to.

5.3.1 Simplification

As this algorithm aims to compare the vectors between inflexion points and turning points, the first objective is to find these points. Although the expected starting point in chapter 4 was skeletonisation, this is not necessary for this algorithm, as the drawn stroke is directly stored as an array and, therefore, already a skeleton of the stroke. However, the stroke still needed to be simplified. The first approach used a simplification algorithm, believing a stroke could be simplified to its most defining points if the tolerance was set at the right value. Thus, the Ramer-Douglas-Peucker algorithm was implemented [36]. This algorithm checks if a point is within the specified tolerance between the next two points. Unfortunately, this algorithm works linearly: it starts at the beginning and starts analysing points until reaching the end. The desired behaviour is to start by finding the most defining points and simplifying from there.

With this in mind, another algorithm had to be found. An algorithm that meets these criteria is the algorithm from Sezgin *et al.* [20], used in the algorithms from Taele and Hammond [15] and Taele and Hammond [22]. This algorithm starts by finding definitive points and then adds points till the shape is correctly simplified. The algorithm from Sezgin *et al.* continues by matching curves to the stroke and "beautifying" the result, which was deemed unnecessary for this project and therefore omitted. The first step of their algorithm is to find and filter speed and curvature data from the original stroke. When the user writes on the canvas, an array of points is loaded into the JavaScript. During the collection of these points, the timestamp of each point is stored in a second array. This array is used to calculate the speed. Sezgin *et al.* state that people slow down when writing a turning point. Sezgin *et al.* start calculating a threshold by taking the average of the speed and multiplying it by 0.9. Minima in speed below this threshold are found and stored as possible candidates to be a turning point. However, if the user writes two turning points without speeding up in between, this method will only find one turning point instead of two (see Figure 5.5a).



(a) Input, 63 points

(b) Using speed data, 4 vertices

(c) Using curvature data, 7 vertices

Figure 5.5: Average-based filtering using speed data misses a vertex. The curvature fit detects the missed point (along with vertices corresponding to the artefact along the left edge of the rectangle) [20].

Consequentially, a second set of measurements is needed: the curvature. The curvature is the degree to which something is curved and shows where strokes abruptly change direction. In this case, the curvature of a point is calculated by looking at the three points before and the three after this point in the original path. These points are then entered in Equation 5.1. The threshold for the curvature is found by simply taking the average of the values. The points with a high curvature can be found by taking all maxima above the threshold. These maxima are stored in an array as well. Using solely the curvature may lead to additional points if the writing is unsteady (see Figure 5.5b). Thus, the data for the speed and curvature need to be combined. To achieve this, we use the two arrays with all points which are candidates to be a turning point.

$$\text{curvature} = \frac{|(x_2 - x_1)\cdot(y_i - y_1) - (x_i - x_1)\cdot(y_2 - y_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \tag{5.1}$$

where

x_i, y_i are the coordinates of the point,

x_1, y_1 are the coordinates of the point three points before,

x_2, y_2 are the coordinates of the point three points after.

Finding the best fit

The next step is to find a fit of points closest to the original shape while containing the minimal number of points necessary. Sezgin *et al.* found that the points most likely to be a part of this fit, are the filtered points which are both a minima for speed and a maxima for curvature. Thus, Sezgin *et al.* start by finding the intersection between the two found arrays. Some points in the intersection are extremely close to each other, either due to noise in the data or an unsteady hand. These points can be filtered by reducing all points in close proximity to the middle point. The result is a filtered list of points which contains solely turning points. However, this list might not include all turning points and could have missed inflexion points.

```
// Find the certainty for speed points
function findSpeedCertainty(points, maximalSpeed) {
  // Find the speed certainty for each point
  points.forEach(value => {
    value.certainty = 1 - (value.speed/maximalSpeed);
  });

  return points;
}
```

Code Snippet 5.1: Find the speed certainty for points not yet in the fit (JS).

To find additional turning points and inflexion points, the rest of the filtered points from the speed and curvature array can be added to temporary fits, after which the error of each temporary fit can be calculated. This can take quite some time, depending on the number of points you need to analyse. Therefore, the points are rated based on the certainty that they appear in the fit. The speed certainty is found by subtracting the speed at that point divided by the maximal speed from 1 (see Code Snippet 5.1). The curvature can be calculated by calculating the distance between two points and dividing this by the length of the curve between these two points. The code for this function can be found in Code Snippet 5.2.

```javascript
// Find the certainty for curvature points
function findCurvatureCertainty(points, path) {
  // Initialize the variables
  let distance;
  let curveLength;

  // For each point
  points.forEach(value => {
    // find the index in the path
    let index = path.findIndex(p => p === value.point);

    // If the neighborhood fits completely in the array
    if (index - neighborhood >= 0 && index + neighborhood < path.
 length) {
      // find the distance between the first and last point
      distance = calculateDistance(path[index - neighborhood], path[
 index + neighborhood]);

      // and the length of the curve in the neighborhood
      curveLength = 0;
      for (let i = index - neighborhood; i < index + neighborhood -
 1; i++) {
        curveLength += calculateDistance(path[i], path[i + 1]);
      }

      // and calculate and store the certainty
      value.certainty = distance / curveLength;
    } else {
      value.certainty = 0;
    }
  });

  return points;
}
```

Code Snippet 5.2: Find the curvature certainty for points not yet in the fit (JS).

Calulating the error

These certainties can then be normalised and sorted from largest certainty to smallest; afterwards, they are added to a list of temporary fits in consecutive order. That means if point three of a list is added to a temporary fit, points one and two are included too. Technically, a fit could consist of all points in the arrays for speed and certainty; however, due to the relative simplicity of individual kanji strokes, the maximum number of points – that can still be added to the intersection – is set to ten. For each temporary fit, the error is found. The error is found by taking the shortest distance between a point and a line – also called the *perpendicular distance*. This is calculated between each point of the original path and the lines of the temporary fit. This error is then squared, to let small inequalities have a large impact (see Code Snippet 5.3).

```javascript
function findError(path, currentFit) {
  // Initialize the values
  let sumSquaredDistances = 0;

  // For each part of the path
  for (let i = 0; i < path.length; i++) {
    let distance = 0;
    // Square the perpendicular distance to the point
    for (let j = 0; j < currentFit.length - 1; j++) {
      if (currentFit[j].index <= i && i <= currentFit[j + 1].index) {
        distance = perpendicularDistance(path[i], currentFit[j].point, currentFit[j + 1].point);
      }
    }
    sumSquaredDistances += distance * distance;
  }

  // Calculate the error of the point
  let error = sumSquaredDistances / path.length;
  return error;
}
```

Code Snippet 5.3: Find the error for a point of the path to the stroke which contains the current index (JS).

At the end of this function, a list has been compiled, consisting of temporary fits with differing numbers of points and errors. Sezgin *et al.* have determined an error threshold which filters out all errors larger. They continue by choosing the temporary fit with the lowest number of points, stating that the best fit must be the temporary fit with the smallest number of points which has an error below the threshold. Unfortunately, they do not provide the value of this error threshold. The error threshold for this project is set at 200, which seems to simplify most strokes accurately. Admittedly, the result is a little crude sometimes, but the user can rewrite the stroke by pressing the undo button. As this fit replaces the strokes on the SVG canvas, the user can check if the fit is accurate. If not, the user can undo and rewrite the stroke.

However, a small adjustment had to be made to make the fit more accurate. As seen in Figure 5.6, the algorithm acts incorrectly when strokes overlap. The original path is shown in blue in the leftmost image. The indexes of the points of the original path are shown with dots, where the black dots should be in the final fit, and the dark pink dots are handled differently by the algorithms. As can be seen, the algorithm from Sezgin *et al.* does not add points 8 and 10 to the fit. This is caused by two things. Firstly, Sezgin's algorithm matches a point from the original path with *any* stroke in the fit. This means that if the fit consists of points 0, 2, 4, 6, and 12, point 7 is matched to stroke 4 to 6 and point 9 is matched to stroke 2 to 4. This means that point 11 is the only point that increases the calculated error. This leads to the next problem: as the error is relatively small, and the fit with the least points is picked, the algorithm decides to write a stroke from point 6 to point 12 rather than adding point 8 and 10 to the fit. A possible solution would be to lower the error threshold, up until the point that the stroke starts adding point 8 and point 10 into the fit. However, lowering the error threshold leads to a higher number of points in fits without overlapping strokes, as the fits with a small number of points are less accurate compared to the fits with a larger number of points. For this algorithm, we try to simplify the original paths to reduce them to their turning points and inflexion points, meaning the lowest number of points which still accurately describes the stroke. Another solution is to modify the algorithm in such a way that each point of the original path is only compared with the stroke of the fit which envelops the point. This means that the error of point 7 is only calculated if the current line starts at index 6 and ends at index 8, or if the line starts at index 4 and ends at 12. This has the advantage that the stroke fitting is more accurate. Additionally, this causes the fitting to be quicker, as the algorithm does not have to calculate which stroke in the fit is the closest anymore to the points in the original path. The result is an (often) accurate fit containing a minimal number of points with a low error.

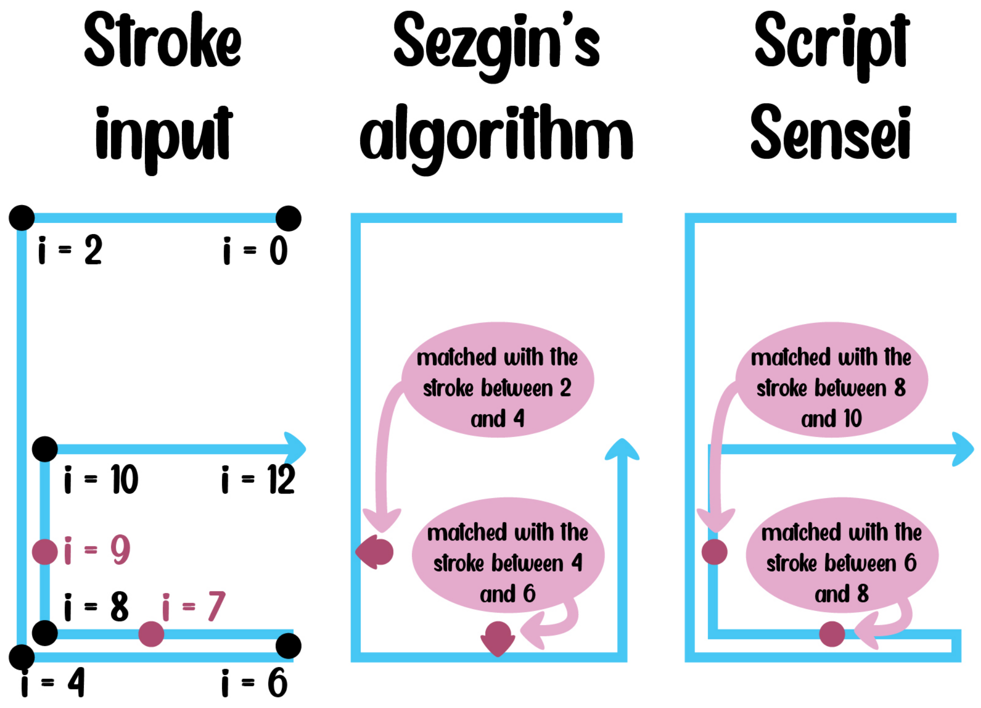

Stroke input
Sezgin's algorithm
Script Sensei
i = 2
i = 0
i = 10
i = 12
i = 9
i = 8
i = 7
i = 4
i = 6
matched with the stroke between 2 and 4
matched with the stroke between 4 and 6
matched with the stroke between 8 and 10
matched with the stroke between 6 and 8

Figure 5.6: Different results of stroke fitting algorithms. The strokes do not overlap for clarity.

5.3.2 Comparison

After finding the fit from the input character, we need to compare the input with the template. With the use of the look-up table from *KanjiVG*, the SVG for the selected kanji is found and the points for the path are extracted. These are then compared, checked, and corrected. The general flow of the algorithm can be seen in Figure 5.7.

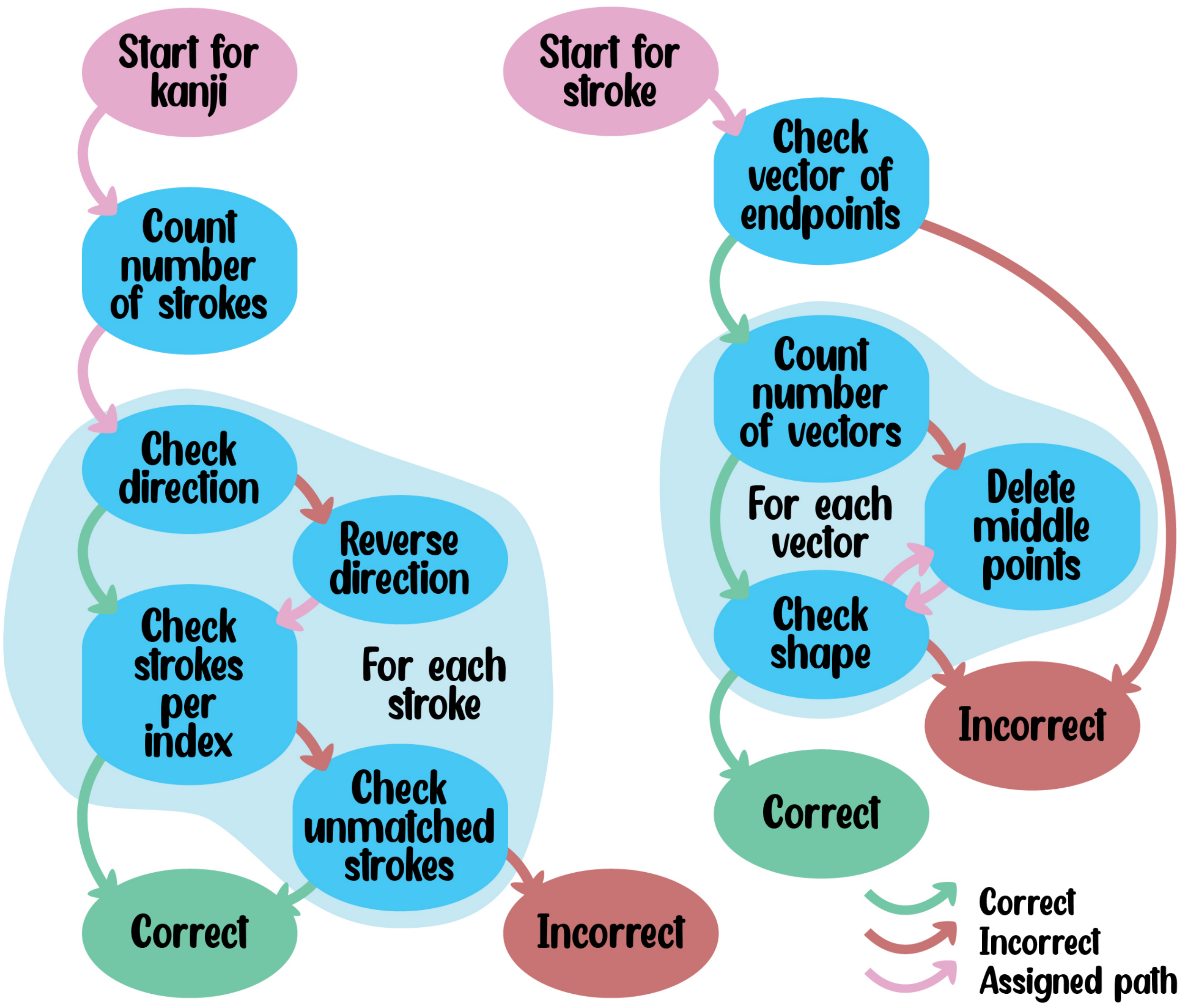


Figure 5.7: The flow of the algorithm.

The algorithm must find mistakes in stroke count, order, direction, shape, size, and location. Classes were made for the kanji and the strokes to keep track of these variables. Integers are initialised as None, booleans are initialised as False, and arrays are initialised empty. The class Kanji consists of the variables:

- integer count: stores the difference in the number of strokes
- array strokes: stores all strokes as Stroke object in the kanji
- array size: stores the averages of the stroke sizes

The Stroke class is initialised during the initialisation of the Kanji class. The Stroke class is initialised with two types of vectors: directional and polar. Directional vectors consist of a displacement in the horizontal and vertical direction. Polar vectors consist of the angle and the length of this movement. The class consists of the following variables:

- integer order: stores the offset in the stroke order
- boolean direction: stores if the direction of the stroke is correct
- boolean size: stores if the relative sizing of the stroke is correct
- integer shape: stores the score for the shape

- integer index: stores the index of the stroke
- boolean reversed_strokes: stores if the stroke is reversed
- array stroke: stores the stroke
- array reversed_stroke: stores the reversed stroke
- array vector_stroke: stores the stroke converted to directional vectors
- array polar_stroke: stores the stroke converted to polar vectors
- array direction_vector: stores the polar vector between the first and last point of the stroke
- array reverse_direction_vector: stores the reversed polar vector between the first and last point of the stroke

These variables are then used to compare the kanji written by the user, the input kanji, with the kanji extracted from the SVG file, the template kanji.

Stroke count

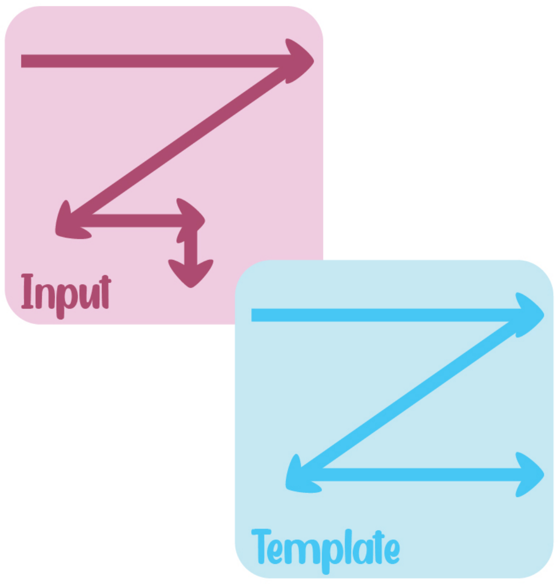
The algorithm starts by comparing the number of strokes of both kanji. The count in the Kanji class is initialized by subtracting the length of the template array from the input array. If the arrays have the same number of strokes, the count will be 0; if the input has fewer strokes than the template, it will be negative; and if the input has more strokes than the template, it will be positive. This way, the stroke count shows if the user wrote too few or too many strokes.

Stroke direction

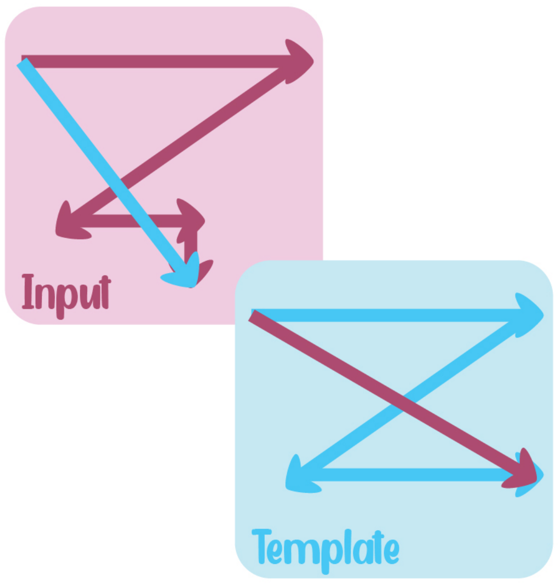
After checking the count, the algorithm starts analysing individual strokes. The first thing that needs to be checked, is the direction. This is achieved by comparing the angle of the polar vectors. If the difference between these angles is less than π, the vectors are in the same direction. If the direction is correct, the direction is set to True. If the direction is incorrect, the stroke is reversed, the boolean reversed_strokes is set to True and all other strokes are reinitialised with the reverse stroke as stroke.

Stroke shape

Subsequently, the shape of the stroke can be checked. The first thing that needs to be verified is whether the direction_vector of the input stroke matches the direction_vector of the template stroke (see Figure 5.8). If the direction between the starting point and the end point is incorrect, the shape will not match either. The stroke is then determined to be incorrect. If the direction_vectors are similar, the next step involves comparing the number of vectors in both strokes. If the strokes consist of the same number of vectors, the angles and lengths of each vector are compared. However, if the strokes do not consist of the same number of vectors, the function delete_points(...) is called. This function creates an array of temporary strokes using points (not vectors), where each stroke consists of the starting and ending point of the original stroke, along with added subsets of points. Stroke objects are then initialised with these temporary strokes. Each of the temporary strokes of the input is compared against the temporary strokes of the template and variables keep track of the longest match and the match with the highest score on the shape. The score of the shape is calculated by dividing the length of the longest match by the length of the original stroke.



(a) The current input and template strokes.



(b) The current input and template strokes with their directional vectors.

Figure 5.8: The directional vector is the vector between the starting and ending points of the input and template strokes. If the directional vector does not match, strokes are not compared to check the shape.

Technically, only the input strokes would have to be sent to the function delete_points(). However, *KanjiVG* contains additional points to their stroke to embellish the physical appearance of the strokes. For example, the kanji one (一) is a horizontal, straight line. Users would write this as a horizontal, straight line. However, in the database, the kanji consists of three points, connected by two curves (see Figure 5.9). This means that the template kanji also has additional points that might need to be removed before the kanji match each other and that each temporary input stroke must be compared to each temporary template stroke. The score of the shape is not influenced by the additional points in the template data, as it is calculated based on the length of the input stroke.

Figure 5.9: A screenshot of the kanji one (一) as found on *KanjiVG* [1].

Stroke order

The algorithm can now recognise strokes, but only compares strokes against strokes with the same index. All strokes which do not have a score for the shape yet, are reset; meaning the integers are set to None, the booleans are set to False, and if reversed_strokes was True, the original stroke is set as the main stroke again. The only variable that is not reset, is the index. Next, all unmatched strokes go through the processes described in "Stroke direction" and "Stroke shape" after which they are reset again. The algorithm starts by taking the unmatched input stroke with the lowest index and finds the best-fitting template stroke to match it. A disadvantage of this approach is that it is comparable to a greedy algorithm: picking the best option for the first stroke, might not lead to the overall best solution. This can be seen in Table 5.1: the best match for input stroke 0 is template stroke 1, leaving input stroke 1 to be matched with template stroke 0. However, for the overall correctness of the algorithm, matching stroke 0 from both kanji with each other leads to a better fit. To be sure the best fit is found, another implementation might be more optimal.

	Template stroke 0	Template stroke 1
Input stroke 0	8	9
Input stroke 1	2	10

Table 5.1: The six types of kanji with examples and effective learning strategies.

Stroke size

The shape of a stroke is deemed correct if the angle and length of each polar vector match (see Stroke shape). However, as the characters are gathered from two different sources – an SVG canvas and an SVG path – the scale of the characters is not equal. Additionally, the user can write the character on the canvas in any size they prefer. Thus, the strokes cannot be scaled by a pre-determined scale. Originally, an approach was tried which takes the bounding box of both shapes. The biggest length from the width and the height was taken for both characters, after which the scaling factor was calculated by dividing the length of the template character by the length of the input character. Then the length of each of the polar vectors from the input character was multiplied by this scaling factor. This approach is rather error-prone. This is illustrated in Figure 5.10. If the user forgets to write the stroke which would increase the size of the bounding box, the scaling of all other strokes in the character would be out of proportion. This would cause these strokes not to be recognised either.

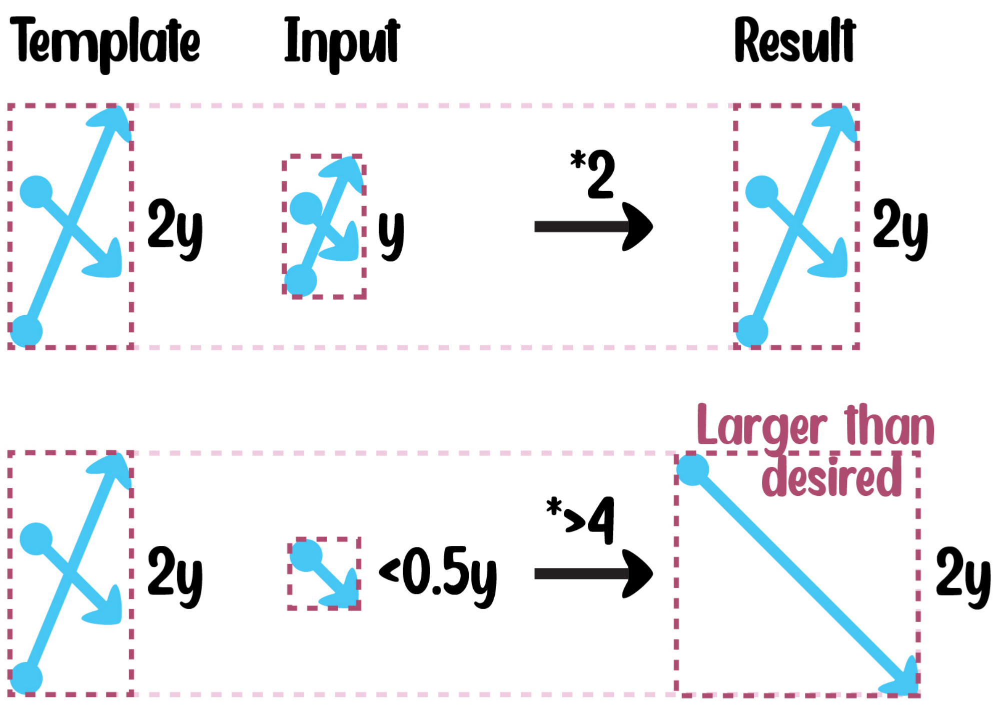


Figure 5.10: The difficulties of scaling based on the bounding box if the input does not match the template.

Thus, another approach was taken. For each vector in the matching input stroke and template strokes the difference in the scale of the strokes was found. Every scaling factor that differed less than the error threshold, would cause the stroke to be labelled as correct (see Code Snippet 5.4). If the shape is deemed as correct, these values would be added to an array, over which the average was calculated. The average was then appended to the size score of the whole kanji.

```
    # Check the size difference purely between matched strokes
    def check_size(input_stroke: Stroke, template_stroke: Stroke):
        global size_tolerance

        size = []
        # For each point in the input stroke
        for point in range(input_stroke.get_stroke_length() - 1):
            input_vector = input_stroke.get_point(point)
            template_vector = template_stroke.get_point(point)

            # Calculate the scale of the strokes
            size.append(abs(input_vector[1] / template_vector[1]))

        if len(size) == 0:
            return 0

        # Calculate the average of the scales
        average = sum(size) / len(size)

        # If there is only one size in the array, it is correctly
   scaled
        if len(size) < 2:
            return average

        # If the scale differs to much from the average
        for point in size:
            if abs(point - average) > size_tolerance:
                # the size is incorrect
                return 0

        # Else, return the average
        return average
    }
```

Code Snippet 5.4: Find if the stroke is scaled correctly (Python).

Stroke scores

After finding the results for each criterion, a final method extracts all variables from the Kanji and Stroke classes. The size still consists of an array of averages to be converted to a single score. To achieve this, the average of the array is calculated. The score of the size can then be found by the number of sizes which differ less than the error threshold divided by the total number of strokes in the array. The values for each criterion are then placed in an array, which is then returned to the writing page, where the feedback is displayed (see subsection 5.2.3).

5.4 Github repository

The code for this project can be found in my GitHub repository. The link is:

https://github.com/EllisDijkstra11/ScriptSensei.git

CHAPTER 6.

Evaluation

To get different perspectives on the website, user evaluations were conducted. Participants received an information letter (see Appendix C.1) and a consent form (see Appendix C.2) upon arrival. After reviewing the documents and signing the consent form, participants interacted with the website and completed a questionnaire. The objective of the user evaluation was to identify which areas of the algorithm for simplification, the algorithm for comparison, and the website are most in need of improvement.

6.1 Procedure

Participants were provided with a laptop displaying the web application and a drawing tablet (see Figure 6.1). First, the participants were asked about any prior experience regarding the use of drawing tablets. If necessary, an explanation of the tablet's usage was provided. To familiarize themselves with the tablet, the participants were instructed to select the kanji using the drawing tablet. Afterwards, an explanation was provided about the objectives of the user evaluation and what should be paid attention to – figuring out the stroke order and direction – when writing the kanji as opposed to the Roman alphabet. The participants were told to pick three kanji to practise writing: one the participants deemed to be easy, one which the participants considered medium, and one the participants found hard. As guidelines, the number of strokes was given for each of the kanji as a reference point. The easy kanji should contain three or fewer strokes, the medium kanji should range between four and six strokes, and the hard kanji should have seven strokes or more.

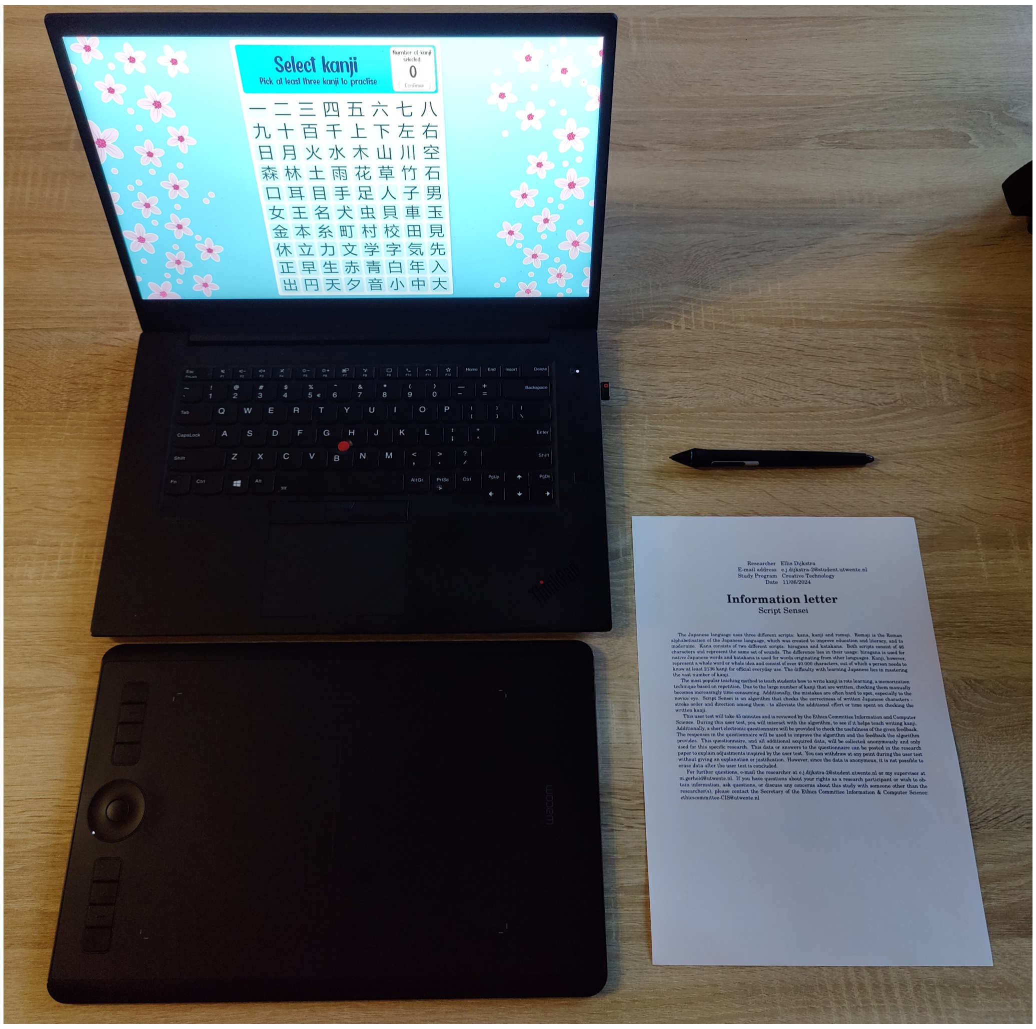


Figure 6.1: The setup used during the user evaluations.

After selecting the kanji, the participants were asked to try to memorize as much as they could from the overview page, before continuing to the writing page. The participants were explained that they would have four tries to write the kanji:

1. Write the kanji from memory.
2. Write the kanji after seeing it on the feedback page.
3. Write the kanji after being shown the right stroke order and direction.
4. Write the kanji with the right stroke order and direction available as hints.

The participants were asked to write the kanji from memory, as their first try is most likely to have the most unexpected input. This allows checking if the algorithm can handle different types of input. Afterwards, the participants could look at how the template kanji on the feedback page. They could use the feedback to gain clues about the stroke order and direction. If the participants still made mistakes in the kanji, the kanji viewer from *KanjiVG* was used to show the correct stroke order and direction. The participants were then asked to close this window and try writing the kanji again. If the participant needed another try, they were asked to write the kanji with the kanji viewer open next to the website. After completing the three kanji the participant selected, the participants were asked to complete the provided questionnaire.

6.2 Results

Six participants partook in the user evaluation. The user evaluation resulted in the answers to the questionnaire and visual observations (see Appendix C.4). Based on the results, the participants were satisfied with the usability and performance of the website and were moderately positive about the appearance of the website. A participant suggested adjusting the hover state of a selected kanji, to make the selection more apparent.

6.2.1 Input device

Three participants had little to no experience using a drawing tablet. Although two participants could figure out relatively easily how the drawing tablet worked, they did have some trouble with unintentionally selecting the text on the webpage, which caused the website to try to move the text rather than allowing the participants to write the kanji. The pen of this drawing tablet also had buttons on the side, which were programmed to perform a right mouse click. Clicking this button caused the eventhandler – which checked if the pen was lifted – to stop functioning, which in turn caused the participant to write a continuous stroke until they moved the cursor out of the canvas. However, the participants could undo this stroke and continue their previous progress. One of the participants also suggested a larger canvas due to trying to write outside the canvas, but this could also be solved by an increased familiarity with the drawing tablet. The third participant without experience with a drawing tablet got so frustrated they preferred to use the laptop's trackpad instead for the last kanji. This did show that using the trackpad did not hinder the participant in writing the kanji. As the simplification filtered any shaking in the lines, the participant could write the medium kanji correctly in one go. The participant answered in the questionnaire that they would have preferred the drawing tablet to have a screen, to make it easier to see where on the screen the tablet is writing. This is an expected problem that comes with the learning curve of a drawing tablet and becomes easier after a little practice.

6.2.2 Simplification

Another point of frustration for one participant was a low score on the shape of a kanji if they deemed the stroke to be correct. The participant offered a solution in the form of showing the turning points and inflexion points in the fit as dots. Other participants expressed vocal annoyance about the disappearance of the original stroke and the replacement by the fit, which would suggest that displaying the points of the fit as dots could be a good solution. Due to the relative simplicity of individual strokes, participants should be able to keep track of how the points are connected. The participants would, however, need to be aware of how the points should be interpreted.

6.2.3 Feedback

There was quite a lot of feedback about the current approach to display the provided feedback. Although the participants agreed that the website could help with learning kanji, they could not interpret the feedback well enough to apply it in the next try for a kanji. Suggestions to improve the feedback were to create a help button to explain each type of error, preferably with images to illustrate how the feedback works. Another suggestion was to show the written kanji next to the template, allowing the participants to compare the kanji visually. Two participants suggested making the incorrect strokes in the written kanji red, to show which strokes should be improved most. Something that would help a participant, would be to show the stroke order and direction with which the kanji was written. One participant remarked that the current methods of displaying non-recognised characters and concatenated strokes could be improved as well.

The usefulness of the feedback per type of error was ordered by finding the average score of the responses to the questionnaire. The participants were instructed to fill in a 1 if they did not realise feedback on this type of error was given. The most useful feedback was the order of the strokes, followed by the direction, shape, and count of strokes. The least useful was the feedback about the size. It was expected that the count would not be very useful, as this was not updated. The difference in the usefulness of the shape, direction, and order is small, although the difference in the size and count is relatively large.

6.2.4 Limitations

After the user evaluations, it became clear that the IDE did not show new updates to the pages, including the label and the number which displayed the count. A participant expressed vocal annoyance about the ambiguity of the label and the inaccuracy of the score.

CHAPTER 7.

Discussion and Future Work

Script Sensei is an acceptable website to evaluate written Japanese characters. The stroke-based algorithm can compare the input kanji with the template kanji correctly; current difficulties mostly stem from the algorithm used to find the turning points and inflexion points of the input kanji. Therefore, most suggestions in this section will be focussed on the website and the provided feedback.

Due to the limited time allotted for this project, not all views could be implemented before the deadline. This section provides suggestions for improvements to be made which could enhance the algorithm and website further. These enhancements stem partially from unfinished or unimplemented ideas from chapter 3, unmet specifications from chapter 4 or feedback from the evaluation (see chapter 6).

The biggest suggestion is to refactor the algorithm; instead of comparing strokes by index and then matching the unmatched strokes, each stroke of the input character can be checked against each stroke of the template character. These matches can then be scored, and the match with the highest score is returned. This would require some additional changes, such as making the scores of the elements more descriptive. For example, the current size counts the number of strokes that are scaled correctly; instead, the difference in scaling could be calculated and displayed. This would also remove the need for error thresholds for the angle and the length, as strokes that do not match would get a low score.

Not all requirements from chapter 4 have been met. This is an updated overview of the met and unmet FRs and NFRs.

☑ FR 1. The algorithm *must* check written Japanese characters.
☑ FR 2. The algorithm *must* correct written Japanese characters.
☑ FR 3. The algorithm *must* not give any hints regarding how the kanji *should* be written.
☑ FR 4. The algorithm *must* allow the user to write the whole kanji before checking it.
☑ FR 5. The algorithm *must* allow users to select which kanji they want to practise.
☑ FR 6. The algorithm *must* be able to check the kanji based on stroke order and direction.
☑ FR 7. The algorithm *must* check the kanji independent of the correctness of the size and the positioning of the kanji.
☑ FR 8. The algorithm *must* be able to recognise mouse input.
☑ FR 9. The algorithm *should* be able to check the kanji, even if the kanji is partially correct.
☑ FR 10. The algorithm *should* be able to recognise strokes consisting of both straight lines and curves.
☑ FR 11. The algorithm *should* be able to recognise input from a drawing tablet.
☑ FR 12. The algorithm *could* provide feedback by giving a rating on multiple areas (count, order, direction, shape, size, and location).
☒ FR 13. The algorithm *could* integrate with external databases for additional information about the kanji.
☒ FR 14. The algorithm *could* include a feature to save and review past practice sessions.
☒ FR 15. The algorithm *could* support multiple users with individual progress tracking.

☑ NFR 1. The algorithm *must* be able to expand the number of kanji it supports.
☑ NFR 2. The user interface *should* be easy to navigate.
☑ NFR 3. The algorithm *should* respond to the user in less than five seconds.
☑ NFR 4. The algorithm *should* be compatible with web browsers.
☑ NFR 5. The algorithm *should* give similar feedback to similarly written kanji.
☒ NFR 6. The user interface *could* ensure usability for users with disabilities.

7.1 Unimplemented ideas supported by the evaluation

This section lists all ideas which were not yet implemented but were mentioned by participants of the user evaluation as missing. This includes improvements for the simplification, comparison, and feedback.

7.1.1 Website

The evaluated way to interact with the website is by using a drawing tablet; however, the new participants struggled with its usage. Having to acclimate to a drawing tablet is expected. One of the suggestions was to use a drawing tablet with a screen instead – which would be significantly more expensive. In the interest of usability, better solutions might be to make the website usable on a mobile or tablet or to create a mobile application. Users would only have to pay for a stylus, rather than a whole drawing tablet.

7.1.2 Simplification

As seen in subsection 6.2.2, the simplification does not work perfectly yet. This mostly results in additional points in strokes or incorrectly simplifying complex shapes. The problem this participant experienced has to do with the way the score for shape is calculated. In one of the tries, the participant tried to draw a stroke that should have consisted of two points but was stored as three. When this stroke is compared against a stroke with two points, the middle point is deleted, and the strokes are matched. However, the shape is given a 5 out of 10, as only one vector is in the matched stroke, which originally had a length of two vectors. The reason this stroke is stored as three points instead of two is probably caused by starting with finding the minimal fit. Next, the close points are filtered, and afterwards, the starting and ending points are added. The third point was likely a part of the intersection of the speed and curvature arrays and thus added to the minimal fit. There are no points to filter out and the starting and ending points are added. There are two sides to this problem: the minimum fit is not filtered to check for neighbouring points and the threshold for the curvature and speed might need to be adjusted.

Filtering the minimum fit could be solved by adding the starting and ending points to the minimum fit and removing neighbouring points afterwards. Then, when the temporary fits are found, points can only be added if they are not close to one of the points already in the list or to one of the points already dismissed for being too close.

The other adjustments that could be made to the algorithm for simplification include fine-tuning the values of the multipliers used for the speed, curvature, and error threshold. Right now, the average for speed is multiplied by 0.9 and the average for curvature is multiplied by 1, as suggested by Sezgin *et al.* [20]. However, it was noticed that these values do not provide the optimal number of points. Some of the points from the speed and curvature are too close together. This might be due to some noise in the calculations, which leads to additional minima and maxima. Additionally, the error threshold might need to be adjusted to improve the fits of the input strokes to find the right balance between using the minimal number of points and accurately displaying the simplified stroke.

Another consideration is that the algorithm from Sezgin *et al.* includes the fitting of curves, which was not implemented for this algorithm. This decision was made due to the minimal number of four points for a bezier; however, this approach could be tried to see if this solves the problem. Otherwise, another approach to find the inflexion points on curves could benefit the simplification. Another solution could be implementing the PaleoSketch algorithm (as cited in [22]). The algorithm is sufficient for the difficulty level of the current kanji, but this might prove to be a problem in the future if more complex kanji are added.

7.1.3 Comparison

One participant missed the solution for split and concatenated strokes (see subsection 6.2.3). They suggested making it clearer if strokes are split and concatenated; currently, the algorithm shows that the number of strokes of the input character and the template character do not match, but does not indicate that strokes were split or concatenated. The strokes are not checked for correctness either, as their direction_vector does not match. This improvement would benefit novice learners who have not yet learned which strokes should be concatenated and which should be written individually. Checking if strokes are split or concatenated is a goal for the project which was not met, although it should be relatively easy. To check if a stroke is concatenated, a for-loop can be used to split the stroke at the current point in the index and check the two resulting strokes against unmatched strokes from the template. Technically, all unmatched strokes can be run through this for-loop to see if the split strokes can be matched. If an input stroke was split, a split template stroke should be able to find a match; if an input stroke was concatenated, the split input stroke should be able to find a match. This should work in all cases. Another approach checks if the input character has too few or too many strokes: if there are too few, a stroke is missing or concatenated; if there are too many, a stroke is redundant or split. However, if the user both split and concatenated a stroke, the difference in number of strokes would be zero. Thus, the previous approach should work better.

7.1.4 Feedback

The current feedback page displays the correctly written kanji and a table with the provided feedback. An attempt was made to show both the input kanji and the template kanji next to each other; however, this proved to be difficult, as the canvas was too big to fit on the pop-up in its current size. Resizing the canvas and the written kanji to fit on the pop-up was more difficult than the current timeframe allowed. If implemented, users would be able to visually compare their kanji with the template, which would make it easier to see why certain strokes were or were not recognised. Although there were no answers to the open questions about the size criteria, it was regarded as the least useful of the current metrics. This is most likely caused as the score for the size is displayed for the whole kanji rather than for each stroke. It is difficult to see which strokes are incorrectly sized when the data is generalised for the kanji. This could be implemented by finding the average scale for the kanji. Afterwards, the scale of each vector can be compared against this average.

Another suggestion for the criteria was to make sure that unrecognised strokes are displayed differently than by leaving the score for the shape empty. The current method did not draw enough attention. This could be solved by merging the cells for the stroke and displaying "Stroke not recognised" instead.

7.2 Alternative unimplemented ideas

This section consists of all ideas mentioned in chapter 3 and chapter 4 which could not be executed within the allotted time for the project. However, the participants of the user evaluation did not remark on any of these problems.

7.2.1 Database

There were a few problems with the database *KanjiVG* [1]. In addition to the embellished strokes (see Figure 5.9), the lookup table was not able to find the SVG files for the kanji four (四), five (五), and year (年). Another issue is that not all of the strokes were extracted from the database in the order that they should be written. Although this was not checked for all kanji, it was seen in the kanji life (生). The way this kanji is displayed on the website, the vertical stroke should be written as third; however, in the retrieved array, this stroke is placed last. It is not clear why this happens.

Additional options for the database are to expand the number of available kanji or to integrate other databases (such as *Kanji alive* [35]) to provide additional information about the kanji.

7.2.2 website

Several ideas for features and gamification are described in section 3.3 and section 3.4. Not all of them have been applied to the current website; however, several elements could greatly enhance the user experience. Currently, the website only works for a single session; there are no options to continue a set of kanji in case the user has to leave and there is no progress tracking to see which kanji have been learned so far. However, this could greatly improve the learning experience and usability of the website. Additionally, the user base could be made more accessible for users with disabilities, as currently no measures have been taken that focus on accessibility. Another easy improvement is adjusting the current options on the feedback page. After writing a kanji and checking the feedback, the only current options are to retry writing the kanji or to move on to the next kanji. A third option could be added easily by allowing the user to move on to the next kanji without removing this kanji from the list. This could also help prevent the tip-of-the-pen phenomenon and allows the user to practise kanji again without having to reselect the kanji.

7.2.3 Comparison

This algorithm aimed to find mistakes in stroke count, order, direction, shape, size, and location. The algorithm handled all these criteria, except for one: stroke location. With the current algorithm, the only way to check the relative position of the strokes would be to check the scaled direction_vector between the starting points of both strokes by relating them to a central point. However, finding this central point presents a similar issue to the scaling problem with a bounding box (see Figure 5.10). Another solution would be to calculate the average of all starting and ending points – negating the effect of incorrect fits – but this would also lead to a skewed centre point in the case of an incorrect number of strokes. The suggested solution uses the relative location of the strokes. To check the relative location of strokes, you need to be sure the stroke you are comparing to is correct, which means that this can only happen after a stroke is confirmed. If the alternate approach is executed, the algorithm assumes all strokes are correct, solving this problem. The location could then be calculated by comparing the direction and length of the vector between the start points of the same two strokes in the input and template character.

7.2.4 Feedback

Another possible improvement for the feedback page was to change the background colour of the elements to match if the data was correct or incorrect. Visually, it would be easier to find the points that need improvement if the correct elements were displayed in green and the incorrect ones in red. The elements with a scale could be displayed with a colour between green and red to display where in the scale the mistake falls.

7.3 Alternative ideas from the evaluation

The only suggestions which were not already an idea for this project, are related to the current way feedback is provided.

7.3.1 Feedback

If the input kanji and template kanji are displayed together on the feedback page, another addition could be to display the stroke order and the stroke direction in the written kanji. The user could forget the order in which they wrote the strokes if the user has to write a kanji with a lot of strokes. This could be achieved by placing the index of the stroke at the starting point of the stroke, which would both show the order and the direction. Another suggestion was to add a help button to explain each metric of the feedback. The participants of the evaluation felt that it would be more clear if they had a better view of what each metric meant, and would prefer an explanation with examples to show how to interpret the feedback.

CHAPTER 8.
Conclusion

Most software currently on the market supplies their users with a template or hints to simplify checking the written kanji. For this purpose, Script Sensei has been developed – a vector-based algorithm designed to check and correct written kanji. This project assessed the possibility of a vector-based algorithm to achieve this objective. This was investigated in multiple steps: finding a method to gather input strokes from the user; reducing the input strokes to turning points and inflexion points; converting the turning points and inflexion points into vectors; comparing the vectors with the templates from the *KanjiVG* database [1]; and finally, displaying the feedback based on count, order, direction, shape and size.

Script Sensei addresses the implicit challenges of learning to write kanji, particularly focusing on the continuous practice of writing kanji and the necessity of revisiting learned kanji to combat the tip-of-the-pen phenomenon [11]. Developed as a solution to alleviate the burden of repetitively checking an increasing number of written kanji, Script Sensei employs best practices from educational software design and builds upon existing algorithms and related work in this field.

To collect user input, a dedicated website was created. This website facilitated the use of a drawing tablet during the user evaluations, keeping the natural feel of a pen. These strokes were then simplified using the algorithm from Sezgin *et al.* [20]. The resulting points were converted to vectors, which were compared with the template characters extracted from the *KanjiVG* database. Scores were calculated based on the count and size of the kanji and the order, direction, and shape of individual strokes. These scores were returned to be displayed as feedback to help users correct their writing. The count was evaluated by showing the difference in the number of strokes between the input kanji and the template kanji; the order was shown as an offset between the position of the stroke and where it should have been; the direction was shown as True or False; and the size and shape were displayed as a score out of 10. The six participants in the user evaluation rated Script Sensei highly, stating that Script Sensei would be able to teach them how to write kanji (4.7/5.0). Additionally, they would consider using Script Sensei if they were to start studying kanji (4.0/5.0). Suggestions included improving the accuracy of the simplification and improving the method to display the feedback. However, Script Sensei is a suitable alternative aimed at learning to write kanji correctly. With its innovative approach, Script Sensei showcases the potential of vector-based algorithms to enhance the learning experience for learners seeking to master the substantial challenge of writing Japanese kanji.

Acronyms

DTW Dynamic Time Warping. 13, 19

FR Functional Requirement. 28, 29, 31, 51

JFL Japanese as a Foreign Language. 5, 6, 14, 20

NFR Non-Functional Requirement. 28, 31, 51

Glossary

grapheme The smallest part a kanji can be split into. 8, 15, 21

greedy algorithm An algorithm that picks the most beneficial option by considering the current possibilities without considering future possibilities. 13, 44

hiragana A syllabic Japanese script, used for authentic Japanese words. 5, 12, 14, 15

inflexion point The point on a curved line where the bend of the line alters its direction. 2, 13, 19, 21, 35–37, 39, 49, 51, 52, 56

kana The syllabic Japanese scripts, hiragana and katakana. 5, 6, 8, 15

kanji A morphographic Japanese script, used for (parts of) words. i, v, vi, viii, x–xiii, 1, 2, 5–9, 12–26, 29, 31–35, 38, 40–45, 47–49, 51–54, 56, 57

katakana A syllabic Japanese script, used for loanwords. 5, 15

mnemonic A story to make something more easy to remember. 8, 14

morphographic script A script that uses their characters to represent whole words. 7, 9

perpendicular distance The shortest distance between a point and a line. 38

recognition rate The percentage of characters an algorithm can recognise successfully. 12–14

romaji The alphabetisation of Japanese characters. 5, 9

skeletonisation The reduction of a stroke to the thinnest possible line. 12, 13, 21, 35

stakeholder (A group of) people who are interested or invested in the process and/or outcome of a project. 20

tip-of-the-pen phenomenon The phenomenon of being unable to produce a kanji due to it being on the fringe of recollection [11]. 14, 15, 20, 33, 34, 54, 56

turning point The point that connects a straight line to another line. 2, 13, 19, 35–37, 39, 49, 51, 56

Bibliography

[1] U. Apel, *KanjiVG*, 2009. [Online]. Available: `https://kanjivg.tagaini.net/index.html` (visited on 20/06/2024).

[2] H. Rose, *The Japanese Writing System: Challenges, Strategies and Self-regulation for Learning Kanji*, en. Multilingual Matters, Jun. 2017, ISBN: 978-1-78309-814-9. DOI: `10.21832/ROSE8156`. [Online]. Available: `https://www.multilingual-matters.com/page/detail/?K=9781783098149` (visited on 18/03/2024).

[3] H. Rose, 'Unique challenges of learning to write in the Japanese writing system,' in *L2 Writing Beyond English*, Journal Abbreviation: L2 Writing Beyond English, May 2019, pp. 78–94, ISBN: 978-1-78892-313-2. DOI: `10.21832/9781788923132-008`.

[4] M. Mazzotta and D. L. Chiesa, 'The Role of Learner Affect in L2 Japanese Writing Tutorials,' en, in *12. The Role of Learner Affect in L2 Japanese Writing Tutorials*, Multilingual Matters, Apr. 2019, pp. 215–235, ISBN: 978-1-78892-313-2. DOI: `10.21832/9781788923132-015`. [Online]. Available: `https://www.degruyter.com/document/doi/10.21832/9781788923132-015/html` (visited on 11/03/2024).

[5] K. Ishida and J. Shin, 'Educational Effect of Kanji Learning System,' en, in *2012 IEEE Asia-Pacific Services Computing Conference*, Guilin, China: IEEE, Dec. 2012, pp. 211–216, ISBN: 978-1-4673-4825-6. DOI: `10.1109/APSCC.2012.48`. [Online]. Available: `http://ieeexplore.ieee.org/document/6478218/` (visited on 22/03/2024).

[6] J. Kim, 'Analysis of Kanji Learning Strategies Using Strategy Inventory for Learning Kanji (SILK),' en, *jsn Journal*, vol. 8, no. 1, pp. 142–156, Jun. 2018, ISSN: 2586-937X. DOI: `10.14456/jsnjournal.2018.19`. [Online]. Available: `https://so04.tci-thaijo.org/index.php/jsn/article/view/120796` (visited on 21/03/2024).

[7] H. Shimizu and K. E. Green, 'Japanese Language Educators' Strategies for and Attitudes toward Teaching Kanji,' en, *The Modern Language Journal*, vol. 86, no. 2, pp. 227–241, Apr. 2002, ISSN: 0026-7902, 1540-4781. DOI: `10.1111/1540-4781.00146`. [Online]. Available: `https://onlinelibrary.wiley.com/doi/10.1111/1540-4781.00146` (visited on 22/03/2024).

[8] C. T. Nguyen *et al.*, 'Robust and real-time stroke order evaluation using incremental stroke context for learners to write Kanji characters correctly,' en, *Pattern Recognition Letters*, vol. 121, pp. 140–149, Apr. 2019, ISSN: 01678655. DOI: `10.1016/j.patrec.2018.07.025`. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/S0167865518303258` (visited on 09/04/2024).

[9] M. E. Everson, 'Best Practices in Teaching Logographic and Non-Roman Writing Systems to L2 Learners,' en, *Annual Review of Applied Linguistics*, vol. 31, pp. 249–274, Mar. 2011, ISSN: 1471-6356, 0267-1905. DOI: `10.1017/S0267190511000171`. [Online]. Available: `https://www.cambridge.org/core/journals/annual-review-of-applied-linguistics/article/best-practices-in-teaching-logographic-and-nonroman-writing-systems-to-l2-learners/5C173A1955CB587A2808B390BD252DE0` (visited on 07/03/2024).

[10] P. Kandrac, 'Maximizing the Efficiency in the Kanji Learning Task by Multicriteria-Based Ordering,' en, 2020, Publisher: [object Object]. DOI: 10.13140/RG.2.2.11675.62242. [Online]. Available: http://rgdoi.net/10.13140/RG.2.2.11675.62242 (visited on 21/03/2024).

[11] N. Chikamatsu, 'L2 Japanese Kanji Memory and Retrieval: An Experiment on the Tip-of-the-pen (TOP) Phenomenon,' en, in *Chapter 2: L2 Japanese Kanji Memory and Retrieval: An Experiment on the Tip-of-the-pen (TOP) Phenomenon*, Multilingual Matters, May 2005, pp. 71–96, ISBN: 978-1-85359-795-4. DOI: 10.21832/9781853597954-004. [Online]. Available: https://www.degruyter.com/document/doi/10.21832/9781853597954-004/html (visited on 22/03/2024).

[12] *Getting Started with Kanji*, en, Nov. 2018. [Online]. Available: https://www.nippon.com/en/views/b05605/getting-started-with-kanji.html (visited on 22/03/2024).

[13] C. I. Hitosugi and R. R. Day, 'Extensive reading in Japanese,' en, vol. Reading in a Foreign Language, no. 16, pp. 20–30, Apr. 2004, ISSN: 1539-0578.

[14] H. Rose and L. Harbon, 'Self-Regulation in Second Language Learning: An Investigation of the Kanji-Learning Task,' en, *Foreign Language Annals*, vol. 46, no. 1, pp. 96–107, 2013, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/flan.12011, ISSN: 1944-9720. DOI: 10.1111/flan.12011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/flan.12011 (visited on 21/03/2024).

[15] P. Taele and T. Hammond, 'Hashigo: A Next-Generation Sketch Interactive System for Japanese Kanji,' en, Jan. 2009.

[16] J. Hamari *et al.*, 'Does Gamification Work? – A Literature Review of Empirical Studies on Gamification,' en, in *2014 47th Hawaii International Conference on System Sciences*, Waikoloa, HI: IEEE, Jan. 2014, pp. 3025–3034, ISBN: 978-1-4799-2504-9. DOI: 10.1109/HICSS.2014.377. [Online]. Available: http://ieeexplore.ieee.org/document/6758978/ (visited on 17/04/2024).

[17] B. Morschheuser *et al.*, 'How to Gamify? A Method For Designing Gamification,' en, 2017. DOI: 10.24251/HICSS.2017.155. [Online]. Available: http://hdl.handle.net/10125/41308 (visited on 18/04/2024).

[18] A. M. Toda *et al.*, 'Analysing gamification elements in educational environments using an existing Gamification taxonomy,' en, *Smart Learning Environments*, vol. 6, no. 1, p. 16, Dec. 2019, ISSN: 2196-7091. DOI: 10.1186/s40561-019-0106-1. [Online]. Available: https://slejournal.springeropen.com/articles/10.1186/s40561-019-0106-1 (visited on 18/04/2024).

[19] K. L. Hartono and J. A. Ginting, 'Implementation of web-based Japanese digital handwriting OCR using chain code and manhattan distance,' en, Bandung, Indonesia, 2023, p. 020 017. DOI: 10.1063/5.0174709. [Online]. Available: http://aip.scitation.org/doi/abs/10.1063/5.0174709 (visited on 06/03/2024).

[20] T. M. Sezgin *et al.*, 'Sketch based interfaces: Early processing for sketch understanding,' en, in *ACM SIGGRAPH 2006 Courses on - SIGGRAPH '06*, Boston, Massachusetts: ACM Press, 2006, p. 22, ISBN: 978-1-59593-364-5. DOI: 10.1145/1185657.1185783. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1185657.1185783 (visited on 18/06/2024).

[21] W. An and C. Li, 'Automatic matching of character strokes for computer-aided Chinese handwriting education,' en, in *Proceeding of the International Conference on e-Education, Entertainment and e-Management*, Bali, Indonesia: IEEE, Dec. 2011, pp. 283–288, ISBN: 978-1-4577-1382-8. DOI: `10.1109/ICeEEM.2011.6137807`. [Online]. Available: `http://ieeexplore.ieee.org/document/6137807/` (visited on 05/04/2024).

[22] P. Taele and T. Hammond, 'LAMPS: A sketch recognition-based teaching tool for Mandarin Phonetic Symbols I,' en, *Journal of Visual Languages & Computing*, vol. 21, no. 2, pp. 109–120, Apr. 2010, ISSN: 1045926X. DOI: `10.1016/j.jvlc.2009.12.004`. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/S1045926X09000809` (visited on 05/04/2024).

[23] K. Kobayashi *et al.*, 'Recognition of handprinted Kanji characters by the stroke matching method,' en, *Pattern Recognition Letters*, vol. 1, no. 5-6, pp. 481–488, Jul. 1983, ISSN: 01678655. DOI: `10.1016/0167-8655(83)90090-9`. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/0167865583900909` (visited on 09/04/2024).

[24] T. Chu *et al.*, 'Supporting Chinese Character Educational Interfaces with Richer Assessment Feedback through Sketch Recognition,' en, May 2018.

[25] Y. Ochi *et al.*, 'JUPITER: A kanji learning environment focusing on a learner's browsing,' en, in *Proceedings. 3rd Asia Pacific Computer Human Interaction (Cat. No.98EX110)*, Shonan Village Center, Japan: IEEE Comput. Soc, 1998, pp. 446–451, ISBN: 978-0-8186-8347-3. DOI: `10.1109/APCHI.1998.704485`. [Online]. Available: `http://ieeexplore.ieee.org/document/704485/` (visited on 22/03/2024).

[26] N. Lin *et al.*, 'A multi-modal mobile device for learning japanese kanji characters through mnemonic stories,' en, in *Proceedings of the 9th international conference on Multimodal interfaces*, Nagoya Aichi Japan: ACM, Nov. 2007, pp. 335–338, ISBN: 978-1-59593-817-6. DOI: `10.1145/1322192.1322250`. [Online]. Available: `https://dl.acm.org/doi/10.1145/1322192.1322250` (visited on 02/04/2024).

[27] D. Berque and H. Chiba, 'Coupled Persuasive Systems: A Case Study in Learning Japanese Characters,' en, in *Learning and Collaboration Technologies*, P. Zaphiris and A. Ioannou, Eds., Cham: Springer International Publishing, 2016, pp. 453–462, ISBN: 978-3-319-39483-1. DOI: `10.1007/978-3-319-39483-1_41`.

[28] Y. Bhattacharya and M. Bhattacharya, 'Work in Progress: Design of a Visual Learning Tool for Japanese,' en, in *Proceedings. Frontiers in Education. 36th Annual Conference*, San Diego, CA, USA: IEEE, 2006, pp. 15–16, ISBN: 978-1-4244-0256-4. DOI: `10.1109/FIE.2006.322563`. [Online]. Available: `http://ieeexplore.ieee.org/document/4117212/` (visited on 22/03/2024).

[29] A. Fathoni and D. Delima, 'Gamification of learning kanji with "Musou Roman" game,' en, in *2016 1st International Conference on Game, Game Art, and Gamification (ICGGAG)*, Jakarta: IEEE, Dec. 2016, pp. 1–3, ISBN: 978-1-5090-5479-4. DOI: `10.1109/ICGGAG.2016.8052664`. [Online]. Available: `https://ieeexplore.ieee.org/document/8052664/` (visited on 22/03/2024).

[30] S. C. Ng *et al.*, 'An Adaptive Mobile Learning Application for Beginners to Learn Fundamental Japanese Language,' en, in *Technology in Education. Transforming Educational Practices with Technology*, K. C. Li *et al.*, Eds., ser. Communications in Computer and Information Science, Berlin, Heidelberg: Springer, 2015, pp. 20–32, ISBN: 978-3-662-46158-7. DOI: `10.1007/978-3-662-46158-7_3`.

[31] P. Taele, J. I. Koh *et al.*, 'Kanji Workbook: A Writing-Based Intelligent Tutoring System for Learning Proper Japanese Kanji Writing Technique with Instructor-Emulated Assessment,' en, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 08, pp. 13 382–13 389, Apr. 2020, ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v34i08.7053. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/7053 (visited on 22/03/2024).

[32] N. Iwayama *et al.*, 'Handwriting-Based Learning Materials on a Tablet PC: A Prototype and Its Practical Studies in an Elementary School,' en, in *Ninth International Workshop on Frontiers in Handwriting Recognition*, Tokyo, Japan: IEEE, 2004, pp. 533–538, ISBN: 978-0-7695-2187-9. DOI: 10.1109/IWFHR.2004.51. [Online]. Available: http://ieeexplore.ieee.org/document/1363966/ (visited on 22/03/2024).

[33] *Flask*. [Online]. Available: https://flask.palletsprojects.com/en/3.0.x/ (visited on 20/06/2024).

[34] *Nonfunctional Requirements: Examples, Types and Approaches*, en, Feb. 2024. [Online]. Available: https://www.altexsoft.com/blog/non-functional-requirements/ (visited on 05/07/2024).

[35] *Kanji alive*. [Online]. Available: https://app.kanjialive.com/search (visited on 20/06/2024).

[36] *Ramer–Douglas–Peucker algorithm*, en, Page Version ID: 1230788851, Jun. 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm&oldid=1230788851 (visited on 30/06/2024).

Appendices

APPENDIX A.
Generative AI

During the preparation of this report, the author used ChatGPT to generate a list of kanji with their respective meaning from *Kanji alive* [35]. ChatGPT was also used to help with debugging. After using this tool/service, the author reviewed and edited the content as needed and the author takes full responsibility for the content of the work.

APPENDIX B. List of kanji on the selection page

```
kanji = [
# Numbers
["一", "one"],
["二", "two"],
["三", "three"],
["四", "four"],
["五", "five"],
["六", "six"],
["七", "seven"],
["八", "eight"],
["九", "nine"],
["十", "ten"],
["百", "hundred"],
["千", "thousand"],

# Directions
["上", "up"],
["下", "below, down"],
["左", "left"],
["右", "right"],

# Natural elements
["日", "day, sun"],
["月", "moon, month"],
["火", "fire"],
["水", "water"],
["木", "tree"],
["山", "mountain"],
["川", "river"],
["空", "sky, empty"],
["森", "forest"],
["林", "grove"],
["土", "soil, earth"],
["雨", "rain"],
["花", "flower"],
["草", "grass"],
["竹", "bamboo"],
["石", "stone"],

# Body parts
["口", "mouth"],
["耳", "ear"],
["目", "eye"],
["手", "hand"],
["足", "foot"],

# People and related concepts
["人", "person"],
["子", "child"],
["男", "man"],
["女", "woman"],
["王", "king"],
["名", "name"],

# Animals
["犬", "dog"],
["虫", "insect"],
["貝", "shellfish"],

# Objects
["車", "car, vehicle"],
["玉", "jewel"],
["金", "gold, money"],
["本", "book"],
["糸", "thread"],

# Locations
["町", "town"],
["村", "village"],
["校", "school"],
["田", "rice field"],

# Abstract concepts
["見", "see"],
["休", "rest"],
["立", "stand"],
["力", "power"],
["文", "sentence"],
["学", "study, learning"],
["字", "character, letter"],
["気", "spirit, mind"],
["先", "previous"],
["正", "correct"],
["早", "early"],
["生", "life, birth"],
["赤", "red"],
["青", "blue"],
["白", "white"],
["年", "year"],
["入", "enter"],
["出", "exit"],
["円", "yen, circle"],
["天", "heaven"],
["夕", "evening"],
["音", "sound"],
["小", "small"],
["中", "middle"],
["大", "big"]
]
```

APPENDIX C.

User Evaluation

C.1 Information letter

Researcher	Ellis Dijkstra
E-mail address	e.j.dijkstra-2@student.utwente.nl
Study Program	Creative Technology
Date	11/06/2024

Information letter

Script Sensei

The Japanese language uses three different scripts: *kana*, *kanji* and *romaji*. Romaji is the Roman alphabetisation of the Japanese language, which was created to modernize and improve education and literacy. Kana consists of two scripts: *hiragana* and *katakana*. Both scripts contain 46 characters and represent the same set of sounds. The difference lies in their usage: hiragana is used for native Japanese words and katakana is used for words originating from other languages. Kanji, however, represent a whole word or whole idea and consists of over 40.000 characters, out of which a person needs to know at least 2136 kanji for official everyday use. The difficulty with learning Japanese lies in mastering the vast number of kanji.
The most popular teaching method to teach students how to write kanji is rote learning, a memorization technique based on repetition. Due to the large number of kanji that have to be written, checking them manually becomes increasingly time-consuming. Additionally, the mistakes are often hard to spot, especially to the novice eye. Script Sensei is an algorithm that checks the correctness of written Japanese characters - stroke order and direction among them - to alleviate the additional effort or time spent on checking the written kanji.
This user test will take 45 minutes and is reviewed by the Ethics Committee Information and Computer Science. During this user test, you will interact with the algorithm, to see if it helps teach writing kanji. Additionally, a short electronic questionnaire will be provided to check the usefulness of the given feedback. The responses in the questionnaire will be used to improve the algorithm and the feedback the algorithm provides. This questionnaire, and all additional acquired data, will be collected anonymously and only used for this specific research. This data or answers to the questionnaire can be posted in the research paper to explain adjustments inspired by the user test. You can withdraw at any point during the user test without giving an explanation or justification. However, since the data is anonymous, it is not possible to erase data after the user test is concluded.
For further questions, e-mail the researcher at e.j.dijkstra-2@student.utwente.nl or her supervisor at m.gerhold@utwente.nl. If you have questions about your rights as a research participant or wish to obtain information, ask questions, or discuss any concerns about this study with someone other than the researcher(s), please contact the Secretary of the Ethics Committee Information & Computer Science: ethicscommittee-CIS@utwente.nl.

C.2 Consent Form

Consent Form for Creative Technology

YOU WILL BE GIVEN A COPY OF THIS INFORMED CONSENT FORM

Please tick the appropriate boxes	**Yes**	**No**
Taking part in the study		
I have read and understood the study information dated 11/06/2024, or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction.	○	○
I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.	○	○
I understand that taking part in the study involves interacting with the algorithm and filling in a questionnaire.	○	○
Use of the information in the study		
I understand that information I provide will be used for the research report.	○	○
I understand that personal information collected about me that can identify me, such as my name, will not be shared beyond the study team.	○	○
I agree that my anonymised information can be quoted in research outputs	○	○

Signatures

________________________	________________________	_________
Name of participant [printed]	Signature	Date

I have provided the information sheet to the potential participant and, to the best of my ability, ensured that the participant understands to what they are freely consenting.

___________________________	________________________	_________
Ellis Dijkstra	Signature	Date

UNIVERSITY OF TWENTE.

C.3 Questionnaire

C.3.1 General

I am interested in learning how to write kanji.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

I tried to look at the feedback to see which mistakes I made.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

I have experience using a drawing tablet.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The application is usable.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The design of the application is attractive.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The input device is fitting.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The input device is practical.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The algorithm is able to recognise my strokes.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The application is able to show the mistakes I made in the kanji.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The application could help me learn kanji.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

I would consider using this application if I would ever start learning kanji.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

Do you have any general feedback about the application?

.

.

.

C.3.2 Performance

The site is quick.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The algorithm is quick.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The site performs as desired.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

C.3.3 Feedback

The feedback on the number of strokes was useful.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The feedback on the size of the kanji was useful.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The feedback on the order of the strokes was useful.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The feedback on the direction of the strokes was useful.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

The feedback on the shape of the strokes was useful.

	1	2	3	4	5	
Strongly disagree	O	O	O	O	O	Strongly agree

Do you have any suggestions to improve these metrics?

.____
.____
.____

Do you have any suggestions for additional metrics?

.____
.____
.____

C.4 Results

This test was conducted with six users. Answers to the open questions in the questionnaire are paraphrased.

C.4.1 The questionnaire

General

I am interested in learning how to write kanji.

	1	2	3	4	5	
Strongly disagree		2	3	1		Strongly agree

I tried to look at the feedback to see which mistakes I made.

	1	2	3	4	5	
Strongly disagree			2	2	2	Strongly agree

I have experience using a drawing tablet.

	1	2	3	4	5	
Strongly disagree	2	1		2	1	Strongly agree

The application is usable.

	1	2	3	4	5	
Strongly disagree				3	3	Strongly agree

The design of the application is attractive.

	1	2	3	4	5	
Strongly disagree		1	2	2	1	Strongly agree

The input device is fitting.

	1	2	3	4	5	
Strongly disagree		1	1	2	2	Strongly agree

The input device is practical.

	1	2	3	4	5	
Strongly disagree	1	2		1	3	Strongly agree

The algorithm is able to recognise my strokes.

	1	2	3	4	5	
Strongly disagree		3		3		Strongly agree

The application is able to show the mistakes I made in the kanji.

	1	2	3	4	5	
Strongly disagree	1	1	2	1	1	Strongly agree

The application could help me learn kanji.

	1	2	3	4	5	
Strongly disagree				2	4	Strongly agree

I would consider using this application if I would ever start learning kanji.

	1	2	3	4	5	
Strongly disagree			2	2	2	Strongly agree

Do you have any general feedback about the application?

- I did not understand the feedback that was given to me, but I did like to draw the kanji. It might have helped to have a help button, which shows what each type of feedback points out.
  An alternate way to display wrong lines is to display them as red.
- I think an input device with a screen could have made the application more easy to use.
- Interesting concept, but in my opinion lacks a good overview of what goes wrong. The provided information is not clear enough to show what needs to be improved.
- The application showed me mistakes that I didn't make and gave me a 5/10 on a straight line for shape. This feels frustrating after you think you nailed the kanji.
  It might have been nicer to display a dot at the places where the algorithm displays a bending point rather than showing the simplified curve.
- Somehow show the stroke order and direction of your attempt in the feedback phase, as (especially in the first one or two attempts with more complicated ones) I'd often forget the order I originally attempted to write the kanji. (Maybe show the attempt in the feedback phase in general, so you can compare).
  The 'canvas' could be a bit larger, as I felt that the input device had a large amount of 'empty' space which I kept trying to use.
  In the selection mode, indicate a selection while the mouse is hovering over it, as I wasn't sure if I had actually selected the kanji before I moved the mouse away which caused confusion at first.

Performance

The site is quick.

	1	2	3	4	5	
Strongly disagree					6	Strongly agree

The algorithm is quick.

	1	2	3	4	5	
Strongly disagree					6	Strongly agree

The site performs as desired.

	1	2	3	4	5	
Strongly disagree				6		Strongly agree

Feedback

The feedback on the number of strokes was useful.

	1	2	3	4	5	
Strongly disagree	2	1	1	1	1	Strongly agree

The feedback on the size of the kanji was useful.

	1	2	3	4	5	
Strongly disagree	1	2	2	1		Strongly agree

The feedback on the order of the strokes was useful.

	1	2	3	4	5	
Strongly disagree		1	1	2	2	Strongly agree

The feedback on the direction of the strokes was useful.

	1	2	3	4	5	
Strongly disagree	1	1		2	2	Strongly agree

The feedback on the shape of the strokes was useful.

	1	2	3	4	5	
Strongly disagree	1		1	4		Strongly agree

Do you have any suggestions to improve these metrics?
- Show the metrics with images/examples
- I think the feedback on the metrics could be a lot better. The metrics are useful, once you know how to interpret them. It would be nice if the application – for instance with the wrong number of strokes – shows how the kanji should be drawn with the strokes, or highlights the strokes you have written in the wrong direction. I think overlaying the feedback over the drawn area near the recognized strokes could help a lot here.
- On stroke order, indicate a non-recognized stroke in a different way than simply giving no number.

Do you have any suggestions for additional metrics?
- Depending on the difficulty of the implementation, indicate if two strokes were mistakenly written as one. (For the kanji 'eye', the outline is two strokes which connect in the example font, and I'm not sure how fast I'd've figured out that I was supposed to write two separate strokes instead of just the one if I didn't see the example.)

C.4.2 Visual observations

The next table shows the number of times the participant had to draw the kanji before the kanji was correct. No participants had to use more than four tries to draw the correct kanji.

	Easy (≤ 3)	Medium (4 - 6)	Hard (≥ 7)
Participant 1	1	3	3
Participant 2	1	4	4
Participant 3	1	3	4
Participant 4	2	4	3
Participant 5	1	3	2
Participant 6	1	4	3

Table C.1: The number of times each participant had to write a kanji before it was written correctly.