Big Data and Cloud Innovation (2019)

ORIGINAL RESEARCH ARTICLE

The Prototype of a Story Book Based Platform for Preschool Programming Education

Chun-Hsiung Tseng, Jia-Rou Lin, Yung-Hui Chen*

Department of Electrical Engineering, Yuan Ze University; E-mail: lendle_tseng@seed.net.tw

ABSTRACT

The importance of learning how to program can never be over-estimated. Even though there are already programming-learning applications for young children, most of the applications in this field are designed for children in the elementary school age or even above. Teaching younger children, for example, preschool kids, how to program appears more challenging. From our survey, it appears that even preschool kids can understand how to do programming and the question is simply which tools to use. Preschool kids usually start their reading from picture books. They learn mathematics, arts, histories, and a lot of knowledge with picture books. The goal of this research is to propose a platform for story tellers, illustrators, and programming education experts to cooperate to build picture books to teach preschool kids how to program. At this very initial stage, the platform is developed as a Web application and hence it can be easily accessed by various devices via Web browsers. The platform consists of a lean story editor, a picture book editor, and a programming concept editor.

KEYWORDS: KidsCoding, Programming, KidsEducation

Received: October 9, 2019; ***Accepted***: December 2, 2019; ***Published online***: December 5, 2019

Citation: Chun-Hsiung Tseng, *et al.* The Prototype of a Story Book Based Platform for Preschool Programming Education. 2019; 3(1): 21-30. DOI: 10.18063/bdci.v3i1.1098

Correspondence to: Yung-Hui Chen, chy@mail.lhu.edu.tw

1. Motivation

Teaching kids programming has become an observable trend and has been observed in many recent researches. Since this is an emerging field, there are still quite a few topics in discussing. However, it appears that topics being discussed have been shifted from whether should we teach kids programming to which tools to use for teaching. Perhaps an issue is that there are simply too many tools focused on this topic, commercially or academically. For example, Scratch, ScratchJr., Logo, and CodeMonkey are some frequently mentioned applications. In addition to tools, there are many case studies to prove the usability and successfulness of these applications. Hence, teaching young children how to program is no longer a problem, but how young? What is the most appropriate age for young children to start learning how to program? The answer is still unknown and deserve more exploration.

Even though there are already programming-learning applications for young children, most of the applications mentioned above are designed for children in the elementary school age or even above. Teaching younger children, for example, preschool kids, how to program appears more challenging. Our survey shows that Scratch and CodeMonkey target children aged around 8 or older and Logo is for children aged around 10. As a result, these three applications may not be feasible for preschool kids due to their complexity and their user interfaces. The only exception in the list is

ScratchJr., which claims itself as being designed for children from 5 to 7 years old. Compared with other applications, ScratchJr. has a relatively simple user interface and covers only a much limited programming concepts. Hence, for preschool kids, it appears that ScratchJr. is an appropriate tool for programming education in the market. In fact, during our market survey, we found some other alternatives such as Lightbot Jr, SpriteBox Coding, and Algorithm City Pro. These applications offers lean user interface and the commands are extremely straightforward. It appears that even preschool kids can understand how to use them. However, programming concepts offered by this type of applications are even more limited than ScratchJr. So we did not include this type of applications in this research. Furthermore, the game-like nature of these applications may be of concern for some parents.

Is ScratchJr. the only choice? Imagine teaching kids under 5 years old programming, do we have alternatives comparable with ScratchJr.? Most of existing solutions are designed as APPs more computer applications. This is convenient and can interest kids to learn how to use digital devices. However, in most kindergartens, teachers do not use APPs as their major tools for teaching. On the other hand, preschool kids usually start their reading from picture books. They learn mathematics, arts, histories, and a lot of knowledge from picture books. This does not mean that using digital devices is not good but the fact may give us a hint that picture books do have their advantages. This makes my research team members and I wonder: is it possible to use picture books as a utility for teaching preschool kids to program? After discussing the idea with some picture book publishers, we soon realize that this is a rather challenging task. In its natural, the task is highly cross-domain. To publish a picture book, a publisher has to cooperate with story tellers and illustrators. For most picture book publishers, programming is a new topic. To add a new topic, domain experts have to join the work. The specialty of programming education makes this a difficult task which requires a lot of communications and negotiations. To reduce the difficulties, we propose a software platform to simplify the cooperation complexities.

The goal of this research is to propose a platform for story tellers, illustrators, and programming education experts to cooperate. At this very initial stage, the platform is developed as a Web application and hence it can be easily accessed by various devices via Web browsers. The platform consists of a lean story editor, a picture book editor, and a programming concept editor. Story tellers use the story editor to author their stories. Once an illustrator find a story suits her/his idea, the illustrator simply pick one story, inherit from the story, and connect pictures with the story. Meanwhile, programming education experts can browse the combined results of stories and pictures and then add appropriate programming education elements to the final picture book. The platform should keep user-friendliness in mind to make its all kinds of users feel at ease. This is very challenging since the all three kinds of users may have extremely different usage patterns. Furthermore, the platform is designed as a open repository for its users to access the needed contents. Contents published to this repository will by default use the Creative Common license which emphasizes on the openness. Imagine a game manufacturer which is preparing for building a programming education game for preschool kids or a researcher who needs some contents to verify whether her/his pedagogy is correct. Contents in this repository will be very useful. In the sections below, we will at first presented the literature survey about the current status of programming education for kids and then give a brief description for the platform proposed in this research.

2. Related works

As pointed out by Clements and Gullo, it has already been proven that learning how to program has positive effects on the development of cognitive style and meta cognitive ability for young children (Clements & Gullo, 1984). In the research work of Duncan and Tanimoto, the issues being discussed shifted from whether should kids be taught about programming to at which age should kids learn how to program (Duncan & Tanimoto, 2014). It appears that the trend is already established. One thing to note is that most, except few existing researches were aimed at kids older than 8 years old. Younger children, kids before 7 years old, behave pretty different. Should we introduce digital game-based learning to younger kids is a even more controversial question. The research of Price, Jewitt, and Crescenzi discussed the effect of iPads in pre-school children's mark making development (Price, Jewitt, & Crescenzi 2015). According to their experiment results, in that way, kids can do more mark making and the results are not completely bad. The hardware tool is one thing, the software tool is another. Papadaki, Kalogiannakis, and Zaranis investigated the effect of using

ScratchJr, which is a special version of Scratch and is designed for younger kids, to teach programming concepts to preschool children (Papadaki, Kalogiannakis, & Zaranis 2016). As the researchers stated: "the teaching of programming and development of fundamental programming concepts at the preschool age has attracted the interest of the educational and scientific community." They performed a small-scale pilot study for the evaluation of ScratchJr and the result was good. Furthermore, Manches and Plowman pointed out that although there has been a proliferation of programming tools designed for kids, the pedagogy to be used is still the most mattered (Manches & Plowman 2017).

Which programming concepts should be taught for kids? According to the research results of Martinez, Gomez, & Benott: "Data show that all students can intuitively learn sequence, conditional, loops and parameters." (Martinez, Gomez, & Benott 2015) Furthermore, as pointed out by Sáez-López., Román-González,& Vázquez-Cano, "visual programming" may be a good approach (Sáez-López., Román-González,& Vázquez-Cano 2016) for teaching these concepts for students at the 5th and the 6th grade. Aivaloglou and Hermans stated that "Block-based programming languages like Scratch, Alice and Blockly are becoming increasingly common as introductory languages in programming education." (Aivaloglou & Hermans 2016) These are definitely visual programming environments. In addition to visual programming, gamification is also considered useful. Prensky stated that gaming is a good opportunity to get kids involved in programming (Prensky 2003). Werner, Denner, & Campe also proposed similar results and they pointed out that programming games are beneficial for assessing the computation learning level of kids (Werner, Denner, & Campe 2015). Corneliussen and Prøitz surveyed voluntary groups teaching children and youth basic computer coding skill and draw the conclusion that "The activities emphasize play, while teaching principles of computer science." (Corneliussen & Prøitz 2016) So, it is possible that the success of gaming based learning environments is rooted at the “play” element. Other case studies also concluded that "play" is a important factor to help kids learn programming.

Manches and Plowman proposed a warning that the current phenomenon led to more attention to the tools than to key questions about pedagogy (Manches & Plowman, 2017). According to Duncan, C., Bell, T., & Tanimoto, "There has been considerable interest in teaching "coding" to primary school aged students, and many creative "Initial Learning Environments" (ILEs) have been released to encourage this."(Duncan, Bell, & Tanimoto 2014) These environments basically fall into two categories: text-based and block-based. Debates about which one is better is not difficult to find (Weintrop & Wilensky, 2017). Logo is definitely a very famous text-based solution (Pea, 1987) but is better suited for elder kids, e.g. kids aged around 10 or even higher. On the other hand, ScratchJr. is a block-based solution and was designed for kids aged from 5-7 (Flannery, Silverman, Kazakoff, Bers, Bontá, and Resnick, 2013). Some products sit between the two edges. Pencil Code is one among them and it allows users to toggle between the two modes (Bau, Bau, Dawson, & Pickens, 2015). Sullivan, Bers, & Mihm proposed the use of KIBO, which is a tool s*et al*lowing young children to become engineers by constructing robots (Sullivan, Bers, & Mihm 2017). The experiment held by Strawhacker and Bers adopted a similar approach but they used the LEGO WeDo robotics (Strawhacker & Bers 2015). Fessakis, Gouli, and Mavroudi used a Logo-based environment on an Interactive White Board (Fessakis, Gouli, & Mavroudi 2013).

3. The proposed platform

In this research, we proposed a picture book based platform for preschool kids coding. The platform provides three types of editing tools, including story editing, drawing editing, and programming concept editing. The design goal of the platform is to build a synergistic bridge between the three editors of story, drawing and programming rather than to build a programmer’s tool. As a result, the platform itself is designed to be as lean as possible. First, there is a search based index page for users to find the desired story. Users simply enter the keywords or the name of the author to search. The search page is shown below:

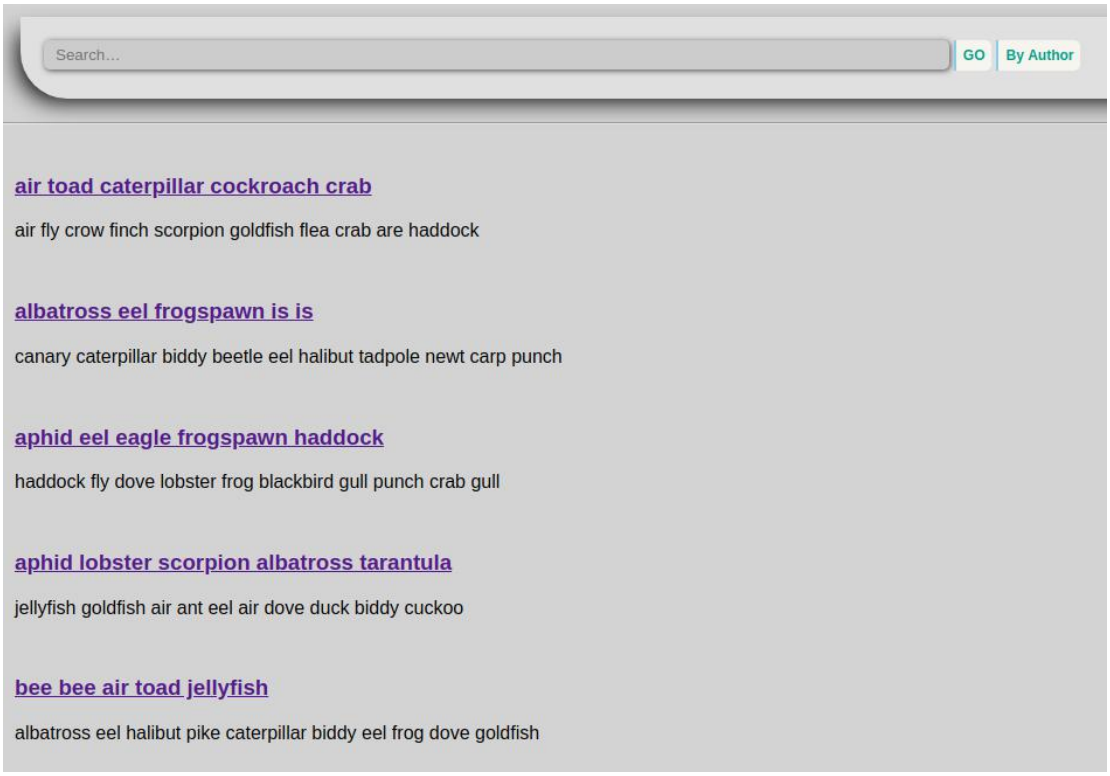

Figure 1. The search page

By clicking the title of the desired story, a page containing the description of the story, the index page, and a list of picture books based on this story will be shown. From this page, a user can view the resulting picture book or creating a new picture book based on the story. The user interface of the index page is illustrated below:

Figure 2. The index page of the selected story

The story editor provides elements including story introduction, role setting, and story description through the platform. The story editing environment is shown in the figure below. It allows story authors to concentrate on story editing. In its user interface, story authors simply choose from two editing modes: dialogues and narrates and then type into the corresponding user interfaces.

Figure 3. The story-editing environment

On the other hand, if the user decides to view the resulting picture book, she/he simply clicks the corresponding icon in the index page. To catch the attention of kids (and to keep it as much fun as possible), a picture is presented in the way of a visual novel game. Viewers simply click on the screen and the dialogues will play automatically. The screenshot of an active picture book is shown below:

Figure 4. An Active Picture Book

Besides, the resulting visual novel game is built based on the monogatari game engine. The platform will generate the required source codes and configurations based on the story and the image resources. The generation process will be described in detail in the next section. Drawing editors adopt the role settings and story description provided by story editors, and then simply connect images with story elements. Program education editors are responsible for adding the appropriate basic programming concepts based on the corresponding story description through the story editor. The editing environment is illustrated in the figure below.

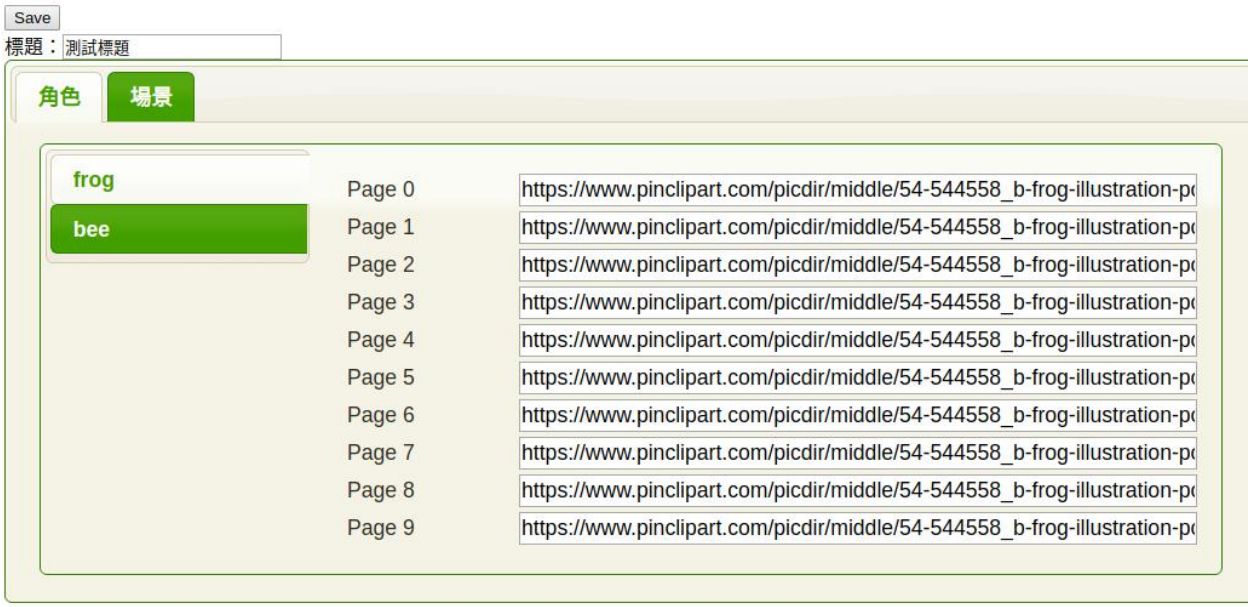

Figure 5. The drawing editor

The work flow of the whole process is listed as the following:

1. story editors create a new story book with introduction and role settings

2. illustrators use the story as a base and create the corresponding images; specific, illustrators create the images for the background and roles of the story book

3. programming educators then use the resulting picture book (completed in step 2) and add programming concepts as narrates

The work flow is depicted below:

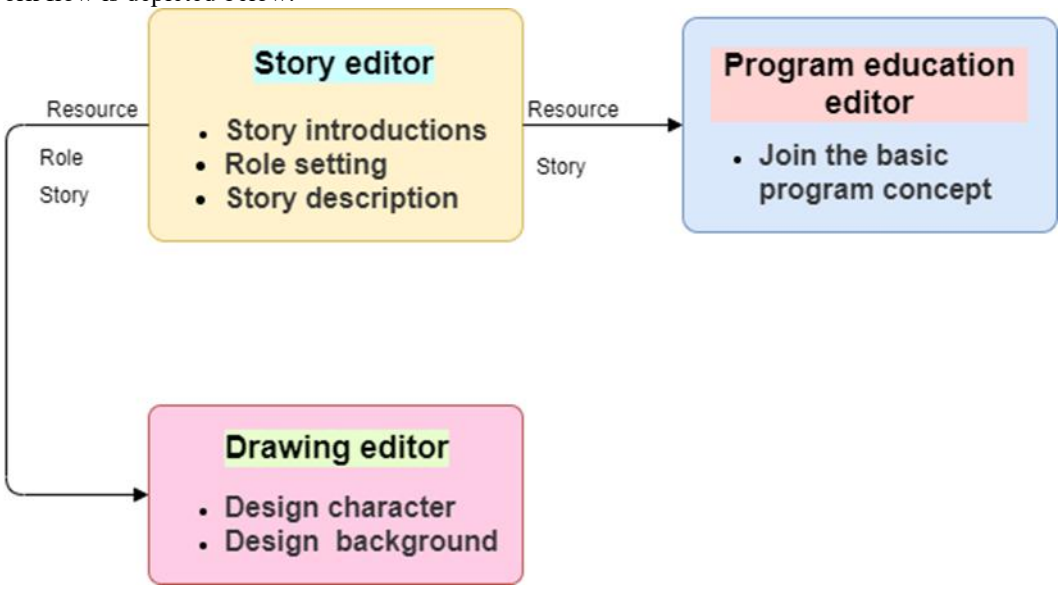

Figure 6. The flow chart

4. System design and implementation of the prototype

The platform is implemented as a Java based Web application and can be divided into 4 parts: the model layer, the service layer, the user interface layer, and the picture book generator. These parts will be introduced in the later subsections.

4.1 The model layer

The model layer encapsulates the data processing logic. There are five major data structures: StoryBookMeta,

StoryBook, PictureBookMeta, SceneSpec, and CharacterSpec. StoryBookMeta holds the meta information, including the title, the author name, the creation date, the last modified date, and the summary, of a story book. In the index page, the platform present only these meta information to users to save bandwidth and processing time. On the other hand, StoryBook holds the full information of a story book. The id and the main content of a story book is stored in the StoryBook data structure. Similar to StoryBookMeta, PictureBookMeta stores only the meta information of a picture book. What is different is that in this platform, a picture book is built upon a story book. As a result, only meta information will be stored for a picture book. The most important meta data of a picture book is a list of the specification of scenes and a list of the specification of characters. The two types of information are stored in SceneSpec and CharacterSpec respectively. These data structures are persisted in a relational database and its schema is shown below:

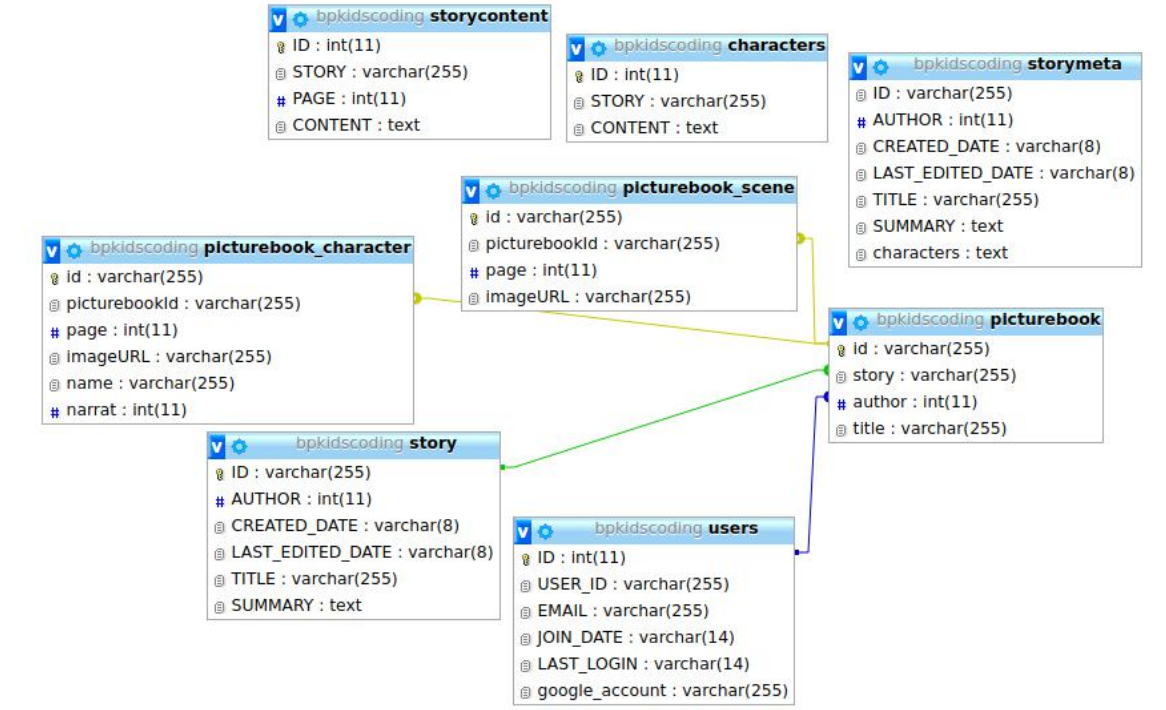

Figure 7: The schema of the underlying database

4.2 The service layer

To lower the coupling, most functionalities of the platform are implemented as restful services. The jersey and spring frameworks are adopted in the design. There are three major services: StoryBookMetaService, StoryBookService, and PictureBookMetaService. The StoryBookMetaService provides mainly the search functions including: findById, findByAuthor, findByKeywords, and findAll. The front-end Web applications rely on these functions to implement the search functionalities. The StoryBookService implements the CRUD for story books. It provides these functions: getStoryBook, addStoryBook, updateStoryBook, and deleteStoryBook. The PictureBookMetaService implements the search and CRUD functions for picture books. It has the following functions: findById, findByStoryBookId, findByAuthorId, addPictureBookMeta, updatePictureBookMeta, and deletePictureBookMeta.

Each of these services adopts the three-tier design paradigm. The three tiers are: the programming interface tier, the implementation tier, and the Web service tier. The programming interface tier declares the signatures of functions with no implementation. The implementation tier implements the corresponding programming interface. The Web service tier exports the corresponding implementation as restful Web services. We use the spring framework to control the dependency injection policy and use the jersey framework to handle the Web service parts. The table below lists the correspondence between functions and Web service endpoints.

Service	Function	EndPoint	Http Command
StoryBookMetaService	findById	/meta/{id}	GET
	findByAuthor	/meta/author/{id}	GET
	findByKeywords	/meta/keyword/{word}	GET
	findAll	/meta	GET
StoryBookService	getStoryBook	/book/{id}	GET
	addStoryBook	/book	POST
	updateStoryBook	/book	PUT
	deleteStoryBook	/book/{id}	DELETE
PictureBookMetaService	findById	/picturebook/{id}	GET
	findByStoryBookId	/picturebook/storybook/{id}	GET
	findByAuthorId	/picturebook//author/{id}	GET
	addPictureBookMeta	/picturebook	POST
	updatePictureBookMeta	/picturebook	PUT
	deletePictureBookMeta	/picturebook/{id}	DELETE

Table 1. The correspondence between functions and web service endpoints

4.3 The user interface layer

The platform itself is presented as a Web application. Most of its user interfaces are implemented with HTML and JavaScript. To make the codes cleaner and more robust, we rely on the Vue.js framework to compose the user interface layer. There are four major pages: the index page, the story book summary page, the story book editing page, the picture book editing page, the picture book viewing page, and the programming concept editing page. The index page allows user to issue search commands to locate the desired story book. Clicking a link in the index page will lead users to the corresponding story book summary page. In the summary page, users can follow a link to the corresponding story book editing page (if the user is the author of the story book) or enter one of the picture book related pages. A picture book page has two modes: the editing mode and the viewing mode. In the editing mode, users associate image resources with story elements. In the viewing mode, users can view the resulting picture book. Finally, the programming concept editing page allows users to associate programming concepts with story elements. Thanks to the coherence of Vue.js, the structure of all these pages are very similar. The workflow of a user interface page is illustrated below:

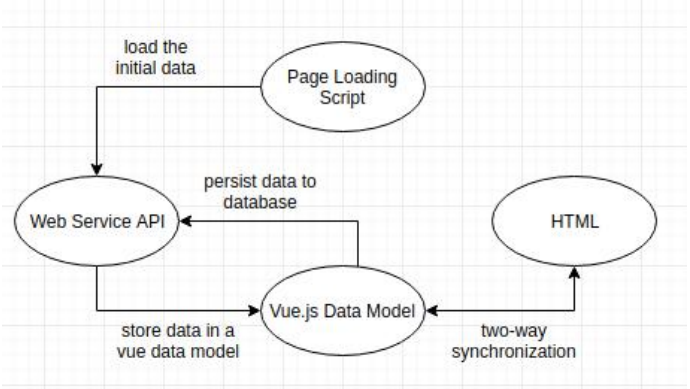

Figure 8. The workflow of a user interface page

4.4 The picture book generator

As mentioned earlier, in this platform, picture books are rendered via the monogatari library, which is a game framework used usually for visual novel games. To ease the rendering task, all picture books are automatically generated. Since monogatari is a JavaScript library, we have to automatically its JavaScript configurations for each picture book. In fact, the Java Server Page technology is adopted to dynamically generate the configuration script. Originally, a monogatari based game should include a configuration script in its header. It will be very tedious if

separated configuration scripts are needed for each picture book. To remedy this, a jsp (Java Server Page) file is used as a pseudo configuration script. The figure below shows the most essential part of the jsp file:

```
<%@page contentType="text/javascript" pageEncoding="UTF-8"%>
<%@page import="org.springframework.web.servlet.support.*" %>
<%@page import="java.util.*, com.google.gson.*" %>
"use strict";
<%
    PictureBookMetaService readableStoryBookMetaService=RequestContextUtils.
            findWebApplicationContext(request).getBean(PictureBookMetaService.class);
    StoryBookService storyBookService=RequestContextUtils.findWebApplicationContext(request).getBean(StoryBookService.class);
    StoryBook storyBook=storyBookService.getStoryBook(request.getParameter("storyBookId"));
    PictureBookMeta meta=readableStoryBookMetaService.findById(request.getParameter("id"));
    MonogatariSession monogatariSession=new MonogatariSession(
            new File(application.getRealPath("/storybook/reader")), storyBook, meta);
    Gson gson=new Gson();
%>
```

Figure 9: The jsp file as a pseudo configuration script

5. Discussion

The platform proposed in this manuscript is different from existing systems and methods in many aspects. First, most existing systems or products, e.g. Scratch and Logo, targets children who are 8 years old or above, and the proposed platform targets children who can read picture books. Usually, kids in Taiwan will read their first picture book at 3 years old or even younger. Furthermore, by utilizing picture books, the proposed method will be more acceptable for parents who concern about the vision health of their kids. Last but not the least, picture books are good medium to promote parent-child reading, and we believe that parents should play an important role in coding education for pre-school kids.

6. Conclusions and future work

In this manuscript, we propose the prototype of a picture book based platform for the programming education of preschool kids. The goal of the platform is to become a cooperation place for story book authors, picture book illustrators, and programming educators. Furthermore, the platform will also serve as a open repository of all the resources needed for teaching programming preschool kids how to program. With such a platform, experts in each field simply focus on her/his own tasks and do not have to worry about other elements of a kids coding picture book. The platform is still in its very initial prototyping stage. In the future, we have set up the following goals to achieve:

1. complete the platform
2. implement connections to social networks so authors can publish their works to social media
3. invite field experts to use the platform and give us feedback
4. try to incorporate more programming concepts into the platform

References

1. Aivaloglou, E., & Hermans, F. (2016, August). How kids code and how we know: An exploratory study on the Scratch repository. In Proceedings of the 2016 ACM Conference on International Computing Education Research (pp. 53-61). ACM.
2. Bau, D., Bau, D. A., Dawson, M., & Pickens, C. (2015, June). Pencil code: block code for a text world. In Proceedings of the 14th International Conference on Interaction Design and Children (pp. 445-448). ACM.
3. Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. Journal of Educational psychology, 76(6), 1051.
4. Corneliussen, H. G., & Prøitz, L. (2016). Kids Code in a rural village in Norway: Could code clubs be a new arena for increasing girls' digital interest and competence?. Information, Communication & Society, 19(1), 95-110. doi: http://dx.doi.org/10.1080/1369118X.2015.1093529
5. Duncan, C., Bell, T., & Tanimoto, S. (2014, November). Should your 8-year-old learn coding?. In Proceedings of the 9th Workshop in Primary and Secondary Computing Education (pp. 60-69). ACM.
6. Duncan, C., Bell, T., & Tanimoto, S. (2014, November). Should your 8-year-old learn coding?. In Proceedings of the 9th Workshop in Primary and Secondary Computing Education (pp. 60-69). ACM.
7. Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. Computers & Education, 63, 87-97.
8. Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013, June). Designing ScratchJr: Support for early childhood learning through computer programming. In Proceedings of the 12th

International Conference on Interaction Design and Children (pp. 1-10). ACM.
9. Manches, A., & Plowman, L. (2017). Computing education in children's early years: A call for debate. British Journal of Educational Technology, 48(1), 191-201.
10. Martinez, C., Gomez, M. J., & Benotti, L. (2015, June). A comparison of preschool and elementary school children learning computer science concepts through a multilanguage robot programming platform. In Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (pp. 159-164). ACM.
11. Papadakis, S., Kalogiannakis, M., & Zaranis, N. (2016). Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: a case study. International Journal of Mobile Learning and Organisation, 10(3), 187-202.
12. Pea, R. D. (1987). Logo programming and problem solving.
13. Prensky, M. (2003). Digital game-based learning. Computers in Entertainment (CIE), 1(1), 21-21.
14. Price, S., Jewitt, C., & Crescenzi, L. (2015). The role of iPads in pre-school children's mark making development. Computers & Education, 87, 131-141. doi: http://dx.doi.org/10.1016/j.compedu.2015.04.003
15. Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. Computers & Education, 97, 129-141.
16. Strawhacker, A., & Bers, M. U. (2015). "I want my robot to look for food": Comparing Kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces. International Journal of Technology and Design Education, 25(3), 293-319.
17. Sullivan, A. A., Bers, M. U., & Mihm, C. (2017). Imagining, Playing, and Coding with KIBO: Using Robotics to Foster Computational Thinking in Young Children. Siu-cheung KONG The Education University of Hong Kong, Hong Kong, 110.
18. Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. ACM Transactions on Computing Education (TOCE), 18(1), 3.
19. Werner, L., Denner, J., & Campe, S. (2015). Children programming games: A strategy for measuring computational learning. ACM Transactions on Computing Education (TOCE), 14(4), 24. doi: http://dx.doi.org/10.1145/2677091.