UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Development of web platform "Story Kids"

Author:
Vladyslav ZADOROZHNYI

Supervisor:
Serhii MISKIV

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

I, Vladyslav ZADOROZHNYI, declare that this thesis titled, “Development of web platform "Story Kids"” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Per aspera ad astra."

Lucius Annaeus Seneca

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Development of web platform "Story Kids"

by Vladyslav ZADOROZHNYI

Abstract

The purpose of this bachelor's thesis is to unite all the available knowledge, experience, and skills to create a web application, which will help children evolve at the same time as entertain through watching educating media content, created by specialists, for setting behavior patterns and developing virtues.

Project resources links:

- Web component repository:
  https://github.com/captainvlad/story_kids.git
- Payment component repository:
  https://github.com/captainvlad/story-kids-payment-server.git
- Released website:
  https://storykids-246f5.web.app/
- Demonstration video:
  https://drive.google.com/file/d/1ZvmJMy-PRC3X_3uHYJ...

Acknowledgements

Primarily, I would like to thank the professors of Ukrainian Catholic University, who have been helping, supporting, and encouraging me since my first day in Computer Science program.

I would also like to express special gratitude to Serhii Miskiv, who influenced this project significantly and made the application better using his experience and knowledge.

Finally, I would like to thank everyone who is involved in the University in all ways, making a huge positive impact on future and present life of the country.

Contents

List of Figures

List of Abbreviations

PC	**P**ersonal **C**omputer
AES	**A**dvanced **E**ncryption **S**tandard
API	**A**pplicable **P**rogramming **I**nterface
MVP	**M**inimum **V**iable **P**rototype
RES	**R**ivest **S**hamir **A**dleman
SDK	**S**oftware **D**evelopment **K**it
SQL	**S**tructured **Q**uery **L**anguage
SSO	**S**ingle **S**ign **O**n
UML	**U**nified **M**odeling **L**anguage
BLoC	**B**usiness **Lo**gic **C**omponent
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol
JSON	**J**ava**S**cript **O**bject **N**otation
REST	**Re**presentational **S**tate **T**ransfer
SAML	**S**ecurity **A**ssertion **M**arkup **L**anguage
NoSQL	**N**ot **o**nly **SQL**

Dedicated to my dear family

Chapter 1

Prerequisites

1.1 General context

This project is a part of a commercial deal made between the author as a developer and executant from one side and the startup team from the other.

- all the presented information in the scope of this document is verified by the client's team, represented by Volodymyr Chernyuk, and thus is not harmful in any way to it.
- some sensitive information about project realization, business goals and business value may not be revealed on the client's demand. For this particular case, there will be a relative informing message.
- a certain part of the work was done by the client's management team. This includes such aspects as customer investigation, prototyping and MVP realization. Despite the fact that I was not particularly related to all mentioned processes (except some discussions which led to minor changes), it is needed to include the description of these modules for a better understanding of project value, philosophy and destination. For this particular case, there will be a relative informing message.

1.2 Naming conventions

The product of this bachelor thesis will be referenced throughout the document by the following names: application, portal, project and website. This is mentioned in order to remove risk of misunderstanding throughout the document.

Chapter 2

Introduction

This chapter is intended to describe entry points of the project development: why this idea was chosen, what are the goals and what impact the application is intended to make, according to its creators' ideas.

2.1 Motivation

As far back as I can remember, I always wanted to make the world a better place to live in. Today's world can be considered as a perfect place for doing this with the use of computer science, its instruments and opportunities.

The reason for this is that digitalization processes expand constantly, quickly and inevitably at the moment: not only teenageers and adults consume resultls of these processes, but all age groups make Internet, smartphones and computers part of their everyday lives. New generation for which Internet is not a new amazing invention, but rather an integral part of the world, was grown and today even the smallest children are not exception for this digitalization tendency.

This is why creating special services for them is not only opportunity, but also a necessity, which was not actual in previous years. More explicitly this thesis will be described below. As of now, the main point is to create an application, which will make a positive impact on society, especially, the youngest part of it.

The main functionality of planned application includes:

- user authentification and login procedure
- subscription creation and managing
- educating video content supply
- serving brief description, which encourages user to watch the video
- providing a brief task in the end of every video

Taking mentioned functionality into account, positive impact on other people's minds is defined as providing an opportunity to get new interesting information and cultivate such character features as curiosity, generosity, kindness and optimism and is the main motivation for me as a collaborator of this project.

2.2 Goals and project value

The main goals are to provide values, which were defined at the stage of project designing and planning. They were divided into three main categories to satisfy: scientific, customer and social.

The categories are following:

- Scientific value

  "Computer science is considered as part of a family of five separate yet interrelated disciplines: computer engineering, computer science, information systems, information technology, and software engineering." - Encyclopædia Britannica online, 2012

  Bearing mentioned quotation in mind, the following conclusion can be announced: a scientific value will be created as part of the work on the project and added to computer science and, more specifically, its software engineering subdivision.

- Customer value

  Customer value can be stated as following: an application, which gives an opportunity of getting interesting information at the same time as discovering something new. The idea is that the videos are designed by specialists in the psychology sphere and are intended to cultivate certain positive virtues and behavior patterns.

  It's necessary to mention that the target audience for the application consists of children of age about 3-7 years, which leads to good chances that the idea and essence of each video will be kept in user's mind, as in this age person is prone to remembering new information closely and forming experience, which will stay for a long time.

- Social value

  Impact, made one certain separate user can be widely multiplied on condition that application is popular and accessible by all means of this term: starting from native language availability and finishing with service's reliability.

  The idea of application is considered to have potential of attracting big number of customers (exact numbers cannot be estimated certain specific research processes taking place). As a result of numerous users, it can be expected that virtues and values, transmitted via the application will spread all around the world and quite possibly that some of them will help to form young user's personality. Multiplying described case, measurable effect on future society will be determined.

One more value, left without a detailed description is business value. It is difficult to estimate at the moment, because of a lack of acquaintance with some aspects, needed for analyzing this area: number of videos planned to add, number of languages, marketing strategy, etc.

Chapter 3

Preparatory work

This chapter is intended to describe preparatory work, made to define the project's perspectives, analyze competitors, briefly discuss strengths and weaknesses of each and outline improvements space, which projected application may be able to fill.

3.1 Competitors analysis

Certain platforms across the internet posses similar functionality as the projected application is going to provide. On the other side, it's possible to define their strengths and weaknesses to understand which new value can be presented to a user and which special benefits can be delivered. It's needed to say that this part of the job was done solely by the client team.

1. Competitors list

Epic: an English-speaking educational platform, intended to provide high-quality books for children and create special closed communities, joined under a special 'class room' concept.

- Link: https://www.getepic.com/
- Strengths:
  - a big number of customers (up to 1 billion reviews yearly)
  - massive various content library (more than 45,000 items)
  - long time on the market and thus, good reputation and respectable image in eyes of newcoming users
- Weaknesses:
  - service is restricted to English-speaking group of users only

Hopster: multilingual application, intended to provide video content, comics, audio and text books.

- Link: https://www.hopster.tv/
- Strengths:
  - free trial period for newcomers
  - a big number of respectable partners, which supply the service with qualitative content to
  - content is available in different languages

  - available both web and apple versions of application

- Weaknesses:

  - content is restricted to relatively small number of categories (geography, alphabet and arithmetic)

Vooks: an English-speaking educational platform, which provides educational video content for children and teachers.

- Link: https://www.vooks.com/
- Strengths:
  - free version for teachers available
  - available on a big number of platforms: TV, web, Android, IOS, desktop
  - certified TV channels, for example, *Discovery Education* and *HappyKids*
  - multilingual applications and web versions
- Weaknesses:
  - content is provided only in one language and is not translatable.

2. Possible improvement space

Possible improvement space consists of solving weaknesses of other services, mentioned before, trying to save alternatives' strengths and trying to position the product in such a way so that many users will see the proposed value and pay for it.

The fact that this application idea has received a grant for realization in Ukrainian Startup Fund program leads to the following conclusion: at the moment, there is a demand for new projects in this sphere and as a result - need of competition in the area is also present.

3.2 Implementation plan

After competitors analysis process was finished, development plan was needed to be created in order to keep track of progress and effective time management. As a result following plan was created:

3.2.1 Planning

This stage includes model (presented by client side) analysis, defining main functionality and surface components projecting.

3.2.2 Researching

This stage includes search of technologies, packages and SDKs, which provide needed functionality and are most convenient to use.

3.2.3 Development

This stage includes all the development process and is the most massive one, consisting of: coding, iterative testing and combining results of the two previous stages.

3.2.4 Initial release

This stage includes performing hosting of the application and expected result is working program, with all the functionality, stated at the very beginning of the plan.

3.2.5 User testing

This stage contains primary user testing, which are intended to be led by the client side team and is intended to highlight all the moments, which should be modified on users' minds.

At the moment of creating this document, the implementation plan is in the very beginning of this stage.

3.2.6 Final release

This stage includes applying changes and fixing problems, which were discovered during user testing and hosting update codebase on the production server.

3.2.7 Application support

This stage includes fixing problems, if there are some applying updates and providing all services stable work.

Chapter 4

Requirements and restrictions

This chapter is intended to clarify expected requirements, state the imposed restrictions and define an acceptance criteria so that the fact of successful implementation of the project can be either recognized or rejected.

4.1 Product requirements

Product requirements were defined in a separate document and were given by the client side. Because of the juridic and commercial importance of the document content, only certain aspects can be described in this thesis in a slightly simplified way.

Product requirements consist of the following points:

- website must be multilingual and support following languages at the beginning: Ukrainian, Russian, English, Spanish and Polish. The multi-language principle must cover the content aspect as well so that content language changes, as well as the website, does.
- website must include the functionality of authentification, creating a subscription and storing certain user information (username, name, surname, subscription plan chosen, etc).
- application must be able to work with following payment systems: Stripe, LiqPay.
- website must be safe and all the stored data must be enciphered before transportation processes.
- application must be available in satisfactory quality in the following browsers: Microsoft Edge, Opera, Mozilla Firefox and Google Chrome.
- application must have an adaptive design that will provide satisfactory quality design on following devices: computer, tablet, mobile.
- website must keep strict structure on all used devices, which includes: content position logic on the screen, screen-filling, navigation logic.
- application architecture must be durable enough to satisfy usage of about 2,000 customers per week.

4.2 Instruments restrictions

From the very beginning, Flutter framework was chosen as the main instrument by the client side. It is condiered to be a good choice for fast, durable and unified development, which also includes web applications development. In order to confirm this opinion, following quotations can be introduced:

"Flutter is an open-source project hosted on GitHub with contributions from Google and the community. Flutter uses Dart, a modern object-oriented language that compiles to native ARM code and production-ready JavaScript code." - Priyanka Tyagi, 2022

"Flutter has benefits that make space for itself, not necessarily by overcoming the other frameworks, but by already being at least on the same level as native frameworks:

- high performance
- full control of the user interface
- dart language
- being backed by Google
- open-source framework
- developer resources and tooling" - Marco L. Napoli, 2020

Together with developer side strengths, chosen framework bears some benefits for the client side:

- a big number of ready-to-use elements makes development process cheaper and faster compared to other alternatives.
- the client side is intended also to expand its applications network on other platforms, mobile platforms, primarily. Taking this into considreation, choosing Flutter as the core technology makes every developer maximally effective and interchangeable, which leads to effective resource management.

Framework was the only instrument that was dictated to be used, in all other aspects instrument choice was made by the developer. More explicit information about chosen instruments, their alternatives, and choosing reasons of them will be given below the document.

4.3 Acceptance criteria

Product requirements document was defined as the most important indicator of the project success. Satisfaction rate of all the essentials stated in this document is defined as the main acceptance criteria.

Chapter 5

Application structure

5.1 Introduction

As the presented application has a complex structure, it is necessary to provide a unified description pattern to make the explanation as understandable as possible.

The description pattern is following:

- description will process be run from highest level of abstraction (modules, major components) to lowest (manager, model, provider).
- before describing any component, there will be described stack of used technologies, if there are any.

5.2 General architecture

The whole solution consists of two main components: website and payment, which serves as a separate application, hosted remotely. The reason for creating a payment server as a separate component lies in the following fact: there are no available SDKs to be used in Flutter applications. At the highest level the project structure can be described with the following diagram:

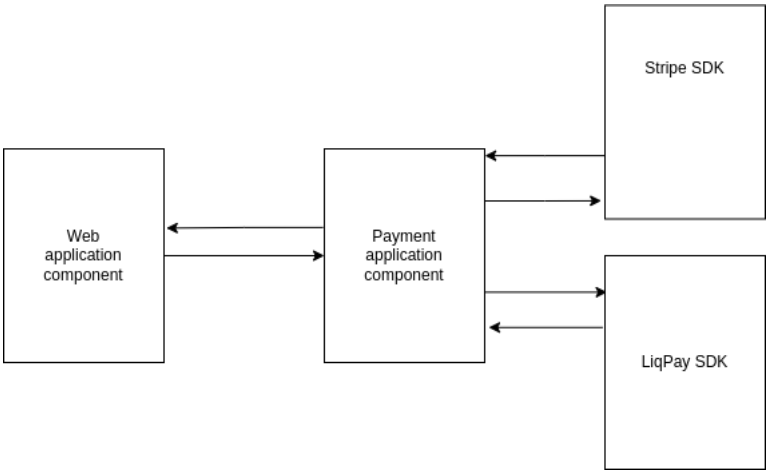

FIGURE 5.1: "Story Kids" platform general architecture

5.3 Payment server component structure

The payment server component is a Flask application, intended to use HTTP requests as communication instrument and exploit payment services APIs in order to

manage processes. In the most general level of abstraction, the component's structure can be presented with the following diagram:

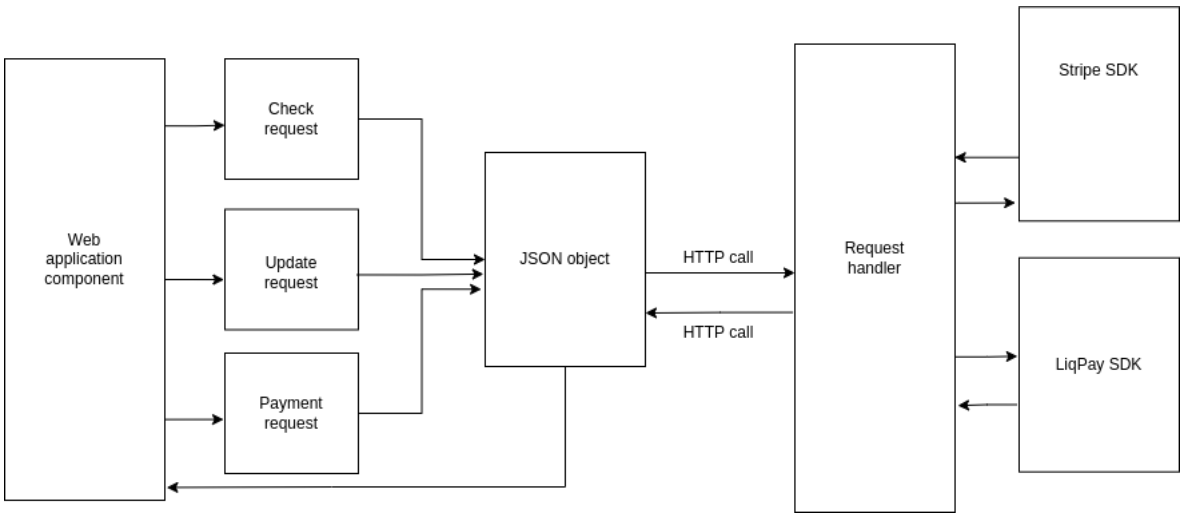

FIGURE 5.2: "Story Kids" payment component structure

5.3.1 Used technologies

1. Google Cloud Run

- **Usage:** component hosting.
- **Benefits:** qualitative support, big number of users, easy deployment.
- **Alternatives:** Amazon Elastic Container Service, Mirantis Kubernetes Engine, SaltStack.

2. Gunicorn

- **Usage:** payment server component concurrent run.
- **Benefits:** easy setting process with Flask app and Google Cloud Run.
- **Alternatives:** uWSGI, Apache Tomcat, Waitress.

3. Flask

- **Usage:** carcass for payment server component logic
- **Benefits:** fast generation, easy development process
- **Alternatives:** Django

5.4 Website component structure

This chapter is intended to give an understanding of web application component structure, main components classes, used approaches, and behavior of all the created items. All the codebase will be split into sub-areas, each of which will be described explicitly in its own subchapter.

To begin with, the project's files tree visualization can be presented with the following diagram:

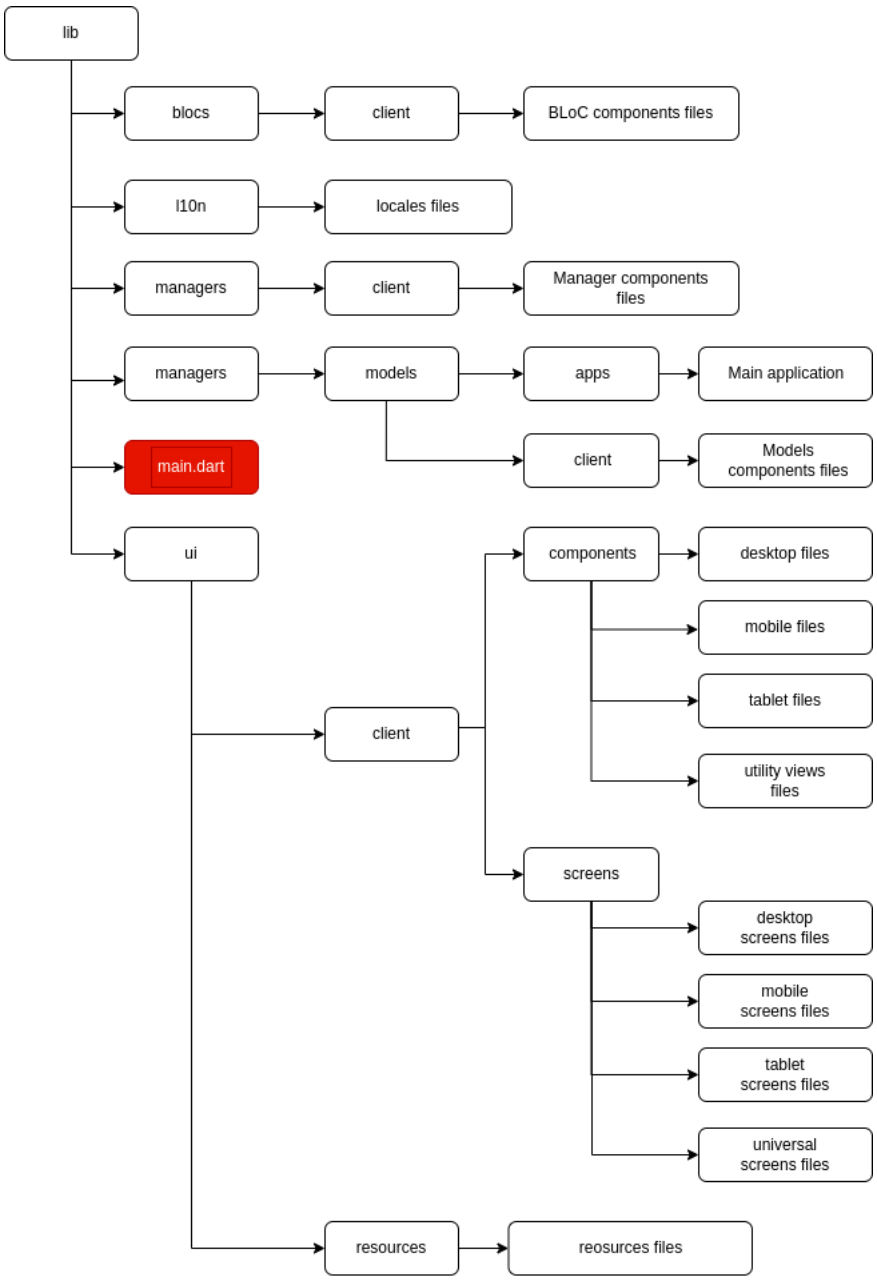

FIGURE 5.3: "Story Kids" web component file tree

Codebase subareas are:

- **User interface:** includes screen files, their components and related aspects.
- **Interim abstractions:** includes models, created to operate with and tools, used for localization implementation.
- **Backend:** includes managers and content providers, created to operate backend services and manage related processes.

5.5 User interface

This section is intended to describe all the components, practices, instruments and approaches, used for implementing user interface subchapter of web application. Explaining the organization of this aspect will be useful for project understanding.

To start with, all the created widgets form following user flow:

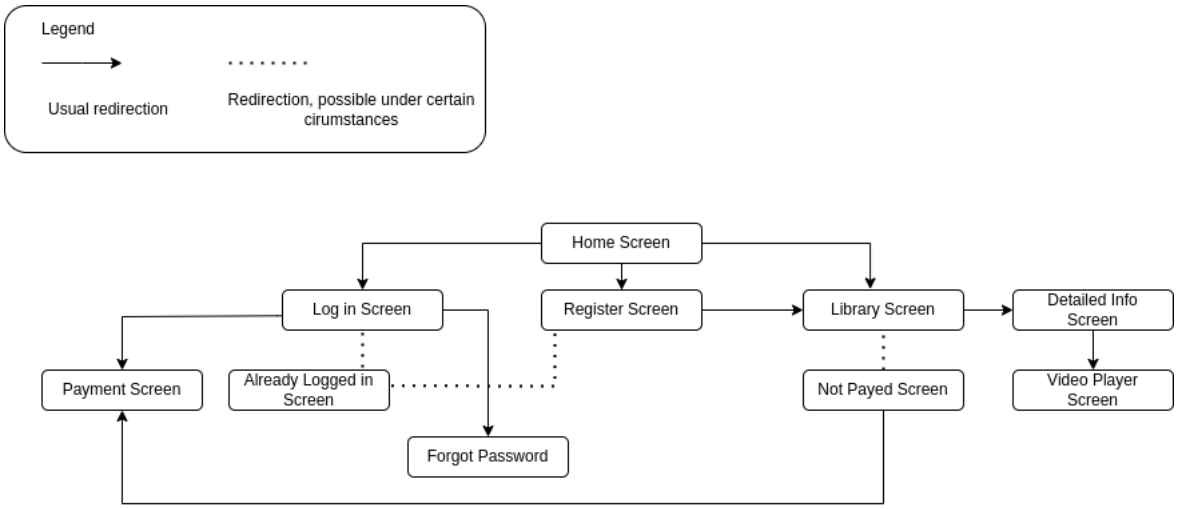

FIGURE 5.4: "Story Kids" user flow structure

5.5.1 Conventions

On all the scope of application following usage rules are spread:

- When any page is reloaded, website navigation is reset to Home Screen. This is done in order to reduce the risk of hacking the service and may be changed in case user testing will give negative feedback in this aspect.
- When trying to access a specific website page, except home screen, website navigation is reset to Home Screen. Reasons for doing this and the possibility of modifying this aspect are the same as for the previous point.
- When user changes language, website navigation is routed to home screen and previous screens are popped out of navigation stack. This is done in order to re-load website content: media and static strings.

5.5.2 Detailed structure

To implement adaptive design and provide valid rendering on different screen types, responsive_builder package was used. Every screen consists of two parts: (header and body) and has three variants (for desktop, tablet and mobile screens). Some screen components are same on mobile and tablet screens, so they are used from same files on mobile and tablet screens. Process of screen displaying is shown in following diagram:

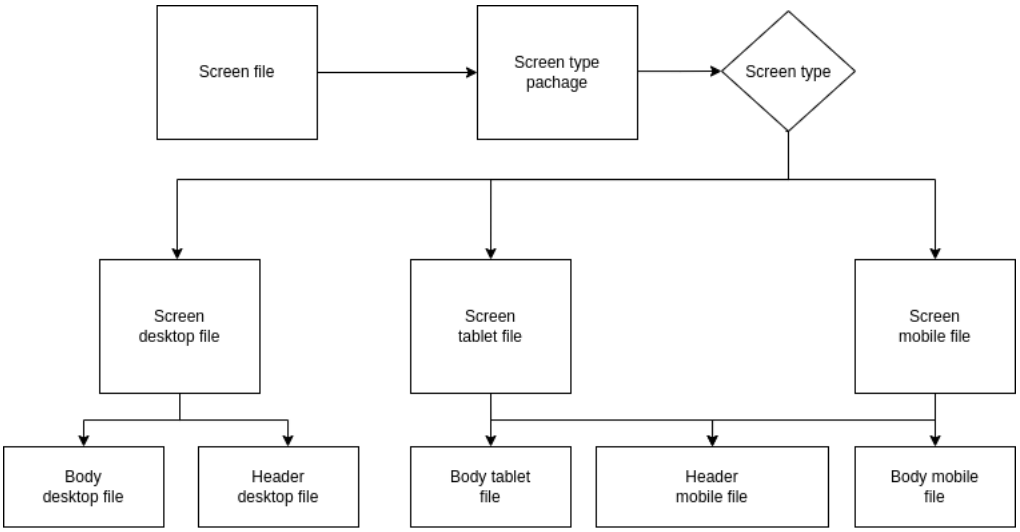

FIGURE 5.5: "Story Kids" screen displaying diagram

5.5.3 Screens roles

1. Already logged screen: accessed when user is already authenticated and tries to log in again or register.

2. Detailed information screen: used for displaying information about chosen content, including description, metadata, authors and illustrations author. The screen takes an argument during initialization, which makes it dependable from the previous screen: Library screen.

3. Failure screen: accessed in case of any operation failure and takes the following arguments (if not passed from previous screens, defined as default): description message and callback when affirmation button is pressed.

4. Forgot password screen: accessed when user tries to restore password, because of losing access to it.

5. Home screen: accessed as application starting point and is the one, where user starts website, redirected to when reloading other screens or trying to access to other screens with special modified link (i.e. https://storykids-246f5.web.app/log_in).

6. Library screen: used to represent library content, consists of carousel view, which represents content and category views, which represent content categories, their names and cards of content, which it belongs to.

7. Log in screen: used to authenticate to already registered user, consists of few input fields, and checkbox tile.

8. Not logged screen: accessed when trying to get access to library screen, being not logged in.

9. Not payed screen: accessed when trying to navigate to the Library screen, being logged in, but having a subscription expired or not paid.

10. Payment screen: accessed after the registration form is filled (in Register Screen) and is intended to get information from user for making payment: credit card data and chosen payment service.

11. Progress screen: used to visualize to user loading process, and has two modes: with header (if content is initialized) or without (otherwise).

12. Register screen: is accessed when creating new account, usually is given when pressing '30 days free' button.

13. Success screen: accessed when any progress process is finished successfully. The screen gets arguments for title, subtitle and confirms button pressed callback.

14. Video player screen: accessed when playback button was pressed (either from Detailed information screen or Library screen).

5.6 Interim abstractions

This section is intended to acquaint the reader with abstractions, created for a better operation process with data. These components' descriptions improve comprehending and understanding of the application's web component.

To start with, all the created abstractions form following structure:

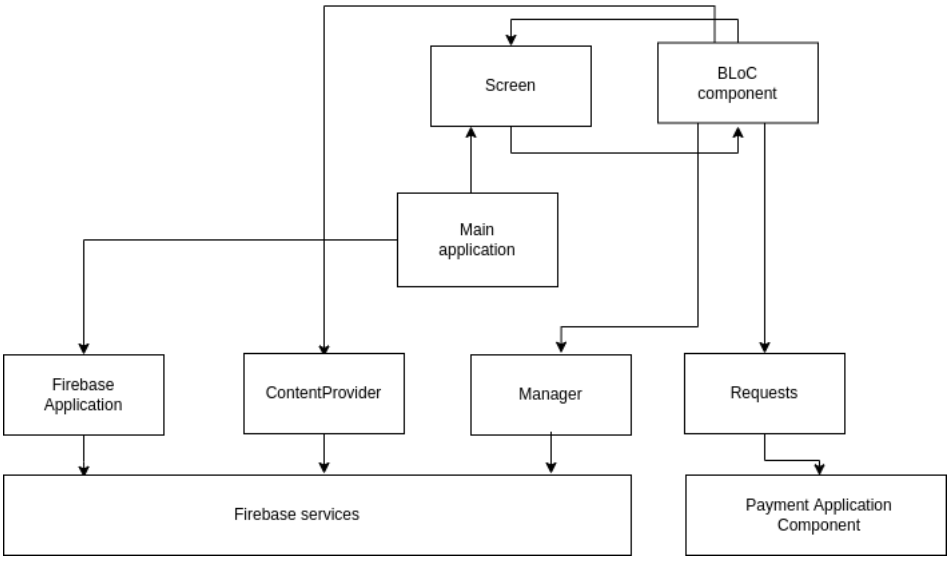

FIGURE 5.6: "Story Kids" backend structure

1. MainApp: application model which provides all the application functionality and states as the entry point of the website.

2. AbstractRequest: base class for describing modified HTTP request, which is made to payment application component.

3. Application (Firebase): abstraction for initializing Firebase application and all its services. Is organized according to Singleton pattern.

4. CheckRequest: implementation of AbstractRequest, intended to make checking requests to payment server component on whether user already has subscriptions.

5. MediaContent: used for storing and manipulating media content related data.

6. PaymentRequest: implementation of AbstractRequest, intended to make payment requests to payment server component

7. Plan: abstraction, used for storing and manipulating with subscription plan related data.

8. UpdateRequest: implementation of AbstractRequest, intended to make checking request to.

9. User: abstraction, used for storing and manipulating with user related data.

10. Utils: used to contain helper methods, which can be used across application

5.7 Backend

As the application is quite complex and demands different problems to be solved (hosting, authentification data storage, etc), a sufficient number of technologies had to be used. When choosing core services to be used, the following factors were the main:

- user friendliness and time of study period
- reasonable price
- high number of supporting materials and large community scale
- reputation of a reliable and safe instrument

Taking following statement into consideration:

"Firebase is a Backend-as-a-Service (BaaS) app development platform that provides hosted backend services such as a realtime database, cloud storage, authentication, crash reporting, machine learning, remote configuration, and hosting for your static files." - Flutter development team, 2022

It can be concluded that Firebase services is the best choice for solving given problems.

One more moment, needed to be described is usage of Singleton pattern, in majority of the described components. Usage of this approach is clearly explained from following statement:

"Use the Singleton pattern when:

- there must be exactly one instance of a class, and it must be accessible to clients from a well-known access point.
- when the sole instance should be extensible by subclassing, and clients should be able to use an extended instance without modifying their code. " - Erich Gamma, 1994

5.7.1 Managers

Here is a brief description pattern, used for explaining role and appointment of each manager:

1. **Usage:** functionality of a manager and main purpose of the item.
2. **Technologies used:** side services, used for solving manager's problem and brief principle of work.
3. **Alternatives:** other technologies, which can satisfy demand and help to solve problem of the component as well.
4. **Benefits:** why chosen technologies were used among the alternatives and describes strengths of the tool.

The managers are following:

1. AuthManager

1. **Usage:** providing log-in and registration processes, organized together with Singleton pattern.
2. **Technologies used:** Firebase Authentication SDK, non-SAML, non-SSO authentification service.
3. **Alternatives:** no ready to use alternatives were provided for solving the problem, except auth package, which is itself a simple superstructure over Firebase Authentication SDK.
4. **Benefits:** plenty of studying material, a big number of users, regarded as a typical choice for solving similar type problems.

2. DownloadManager

1. **Usage:** providing download process from FireBaseStorage and casting raw data to models.
2. **Technologies used:** FirebaseStorage SDK.
3. **Alternatives:** Amazon Storage SDK, sqflite with hosting.
4. **Benefits:** directories and files storage paradigm, easy to use, flexibility, part of Firebase SDK, files access via links.

3. EncryptionManager

1. **Usage:** encryption information before being used in side components
2. **Technologies used:** symmetric AES encryption system with predefined static key, by using encrypt package.
3. **Alternatives:** dbcrypt, rsa_encrypt, simple_rsa2 packages.
4. **Benefits:** simple solution, easy to understand and exploit.

4. InputValidationManager

1. **Usage:** validation input information and generating result and relevant error message depending on special input problem.
2. **Technologies used:** own implementation
3. **Alternatives:** none were found
4. **Benefits:** the only possible solution

5. NavigationManager

1. **Usage:** navigating to screens and providing future screens with needed resources and validity checks before redirecting to other screens
2. **Technologies used:** own implementation
3. **Alternatives:** none were found
4. **Benefits:** the only possible solution

6. NetworkManager

1. **Usage:** making requests to payment server component by executing AbstractRequest implementations (PaymentRequest, UpdateRequest).
2. **Technologies used:** simple HTTP package for making relevant calls was used for solving the problems.
3. **Alternatives:** Dio, Retrofit, Chopper.
4. **Benefits:** most popular and widely used among Flutter developers, designed and implemented by official Flutter team.

7. UiManager

1. **Usage:** gathering together font styles, defining views dimensions and providing valid views output on all screentypes.
2. **Technologies used:** own implementation
3. **Alternatives:** none were found
4. **Benefits:** the only possible solution

8. UserStorageManager

1. **Usage:** storing additional information about the user, which cannot be stored as additional fields in AuthManager. Organized according to Singleton pattern.
2. **Technologies used:** FirebaseDatabase SDK was used for solving the problem.
3. **Alternatives:** same to ones, mentioned before in DownloadManager description.
4. **Benefits:** is NoSQL database, which gives more flexibility in storing data, data stored in simple JSON format.

5.7.2 Content providers

In order to operate downloading process effectively and reduce download time, data segregation process was performed. The principle is following:

1. **'Local' data:** consists of static images, videos, which are used as background elements and form visual style of the website (i.e. main icon on the header). This type of data is not likely to be changed often.
2. **'Remote' data:** consists of media content which forms library and subscription plans. This type of data is expected to be changed from time to time (i.e. when content added or subscription plan was changed).

Content providers use FirebaseStorage SDK as the core technology and are implemented according to Singleton pattern. The providers are following:

1. AbstractContentProvider: used as a basic class for content provider classes and defines main method and fields which are going to be used in classes, which extend it.

2. LocalContentProvider: used for downloading and initializing content which is defined as 'Local' from FirebaseStorage and providing the resources to screens.

3. RemoteContentProvider: used for downloading content which is defined as 'Remote' from FirebaseStorage and providing the resources to screens.

5.7.3 BLoC components

BLoC pattern is the main approach, used for state management inside the web application component. Reasons for using BLoC usage in general are:

- effective state management
- strict code delimitation by its responsibilities on UI and non-UI parts

BLoC components description pattern is following:

- usage
- coverage scope

BLoC components are created with standard Flutter team package, available via link and version 7.x.x was used instead of latest one, as it has more predictable and familiar to most of Flutter developers API, instead of modified yet non-familiar one in 8.x.x version. The components are following:

1. ForgotPasswordBloc

- usage: controlling forgot password input body and transmitting data to relevant manager when processing input data.
- scope: screenBody widget, depending on screen type (ForgetBodyDesktop, ForgetBodyTablet, ForgetBodyMobile).

2. HeaderBloc

- usage: ontrolling user tapping on buttons, located on the desktop and for providing current locale object, which is responsible for localization language.
- scope: whole application.

3. InitBloc

- usage: providing all the needed resources for displaying screen (local, remote and checks if Firebase application is initialized). If not - provides initialization process.
- scope: whole application.

4. LoginBloc

- usage: controlling log in screen input body and transmitting data to relevant manager when processing input data.
- scope: screenBody widget, depending on screen type (LogInBodyDesktop, LogInBodyTablet, LogInBodyMobile).

5. PaymentBloc

- usage: credit card information for managing subscriptions
- scope: screenBody widget, depending on screen type (PaymentBodyDesktop, PaymentBodyTablet, PaymentBodyMobile)

6. InputFieldBloc

- usage: controlling one specific component, control input data visibility.
- scope: one specific widget (accurately: InputCustomField).

7. PlayerBloc

- usage: controlling media content playback process and adjacent operations.
- scope: whole application.

8. RegisterBloc

- usage: controlling registration input body and transmitting data to relevant manager when processing input data.
- scope: screenBody widget, depending on screen type (LogInBodyDesktop, LogInBodyTablet, LogInBodyMobile).

Chapter 6

Testing

This chapter is intended to cover Quality Assurance process, describe its structure and idea. Quality assurance consists of following components:

- unit testing
- integration testing
- manual testing

6.1 Unit testing

Relevant files can be accessed in
test folder of the project. At the moment only one screen is tested in this way because manual and user testing were chosen as primary types of Quality Assurance. The reason for this decision is that after the stage of a testing product via user testing, there is a big probability that certain changes may be applied, which may have a sufficient impact on the application, which in its turn will lead to the necessity of modifying integration tests as well, which will take certain time and resources. Because of this fact, manual testing was given the highest priority over other types.

6.2 Integration testing

Relevant files can be accessed in
test folder of the project. Following tests are designed to cover testing of main aspects of following components: AuthManager, DownloadManager, LocalContentProvider, RemoteContentProvider. As tested components are not so risky to be modified after user testing process, they are bigger and more extensive than unit ones. Each component test has multiple test methods for its functionality.

6.3 Manual testing

This type of testing was chosen as the main direction of testing and is presented by complex of tests, divided by following aspects:

- visual aspect: views displaying, depending on screen type and size
- functional aspect: testing application on the predictability of application work

For visual testing, three virtual devices (via Chrome debug service) with different screen types were used: iPhone 12 Pro, iPad Air and Desktop screens. ALgorithm for the testing was following: display each screen and check for display quality, providing rotation actions.

For functional testing same virtual devices were used as in visual testing. ALgorithm for the testing was the following: display each screen, try all possible variants of actions of usage as a user model: screen redirections, any buttons pressing in any order and any input given to the fields.

Chapter 7

Perspectives

This chapter is intended to describe possible space for improvements, using available version of the application. It will be split on two parts:

- minor changes, which might be applied
- application's possibilities to improve

It is needed to mention that all the stated improvements can (or cannot) be realized due to client side team and due to its vision of the product's future.

7.1 Possible changes

The minor changes, which may help the project develop, are following:

- adding new content to the library.
- adding new subscription plans.
- adding new payment methods for the customers.
- creating bigger number of website localization templates and expanding to new countries.
- fixing current localization strings (as languages strings were not provided by the client side team, Google Translate service had to be used until end of user testing).

7.2 Application's possibilities to improve

Application was planned and implemented that all of the mentioned improvements can be done relatively easy and fast. This is achieved by qualitative services used and opportunities, provided by the core framework Flutter.

Chapter 8

Conclusion

This chapter is intended to describe the progress, made during the time of the application development, outline plans for the project's future and state impact of the project implementation on the author of this thesis (including experience, new knowledge and skills).

In addition, it's needed to mentioned, that some secret information is hidden from public repository and because of this a simply cloned codebase from repository cannot be run locally.

8.1 Progress overview

This document is result of 5 months hard work and resulted in a perspective application which is going to go through certain stages in future. It's needed to say that certain job must be done over the project on its road to final release.

At the moment, the application has passed through initial release stage and is available through the link, published at the very beginning of the document.

During all the implementation process, big amount of information was extracted, processed and used for project realization. Most important is considered to be about:

- full cycle of application development
- web application development
- projecting and implementing own APIs
- direct client and developer interaction
- Firebase SDK services integration into Flutter applications

Although some information from the mentioned topics was known before, this thesis project developed my knowledge, improved my skills and provided very useful experience.

8.2 Secret information

As it was stated before, certain information was hidden from public repository because of its importance. This information consists of:

- configuration data

- enciphering key
- payment application component link

Listed information is available only to client side team and the developer (during the process implementation process).

8.3 Final words

Taking previous facts into consideration, as well as acceptance criteria, stated at the very beginning of the thesis, this job can be regarded as a successfully implemented project.

Bibliography

Alessandro Biessek (2019). *Flutter for Beginners*. Packt Publishing. ISBN: 9781788996082. URL: https://www.amazon.com/Flutter-Beginners-introductory-cross-platform-applications/dp/1788996089.

Encyclopædia Britannica online (2012). *Computer Science*. URL: https://www.britannica.com/science/computer-science (visited on 05/30/2022).

Erich Gamma Richard Helm, Ralph Johnson John Vlissides Grady Booch (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional. ISBN: 8601419047741. URL: https://www.amazon.com/Design-Patterns-Elements-Reusable-Object-Oriented/dp/0201633612.

Flutter development team (2022). *Flutter official documentation*. URL: https://docs.flutter.dev/development/data-and-backend/firebase (visited on 05/30/2022).

Marco L. Napoli (2020). *Beginning Flutter: A Hands On Guide to App Development*. John Wiley Sons, Inc. ISBN: 9781119550822. URL: https://www.amazon.com/Beginning-Flutter-Hands-Guide-Development/dp/1119550823.

Priyanka Tyagi (2022). *Pragmatic Flutter*. CRC Press. ISBN: 9780367612092. URL: https://www.amazon.com/Pragmatic-Flutter-Building-Cross-Platform-Android/dp/0367612097.