

# **Development of a Client-Based Application for Enhancing Pediatric Dental Hygiene through Gamification Strategies**

*Margarita H. Radeva*

*Word Count: 17 023*

A dissertation submitted in partial fulfilment  
of the requirements for the degree of  
**Master of Engineering**  
of the  
**University of Aberdeen.**



Department of Computing Science

April 26, 2024

# Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

A handwritten signature in black ink, appearing to read 'M. Radeva', with a period at the end.

Signed:

*M. Radeva*

Date: April 26, 2024

# Abstract

This thesis presents the development of a novel mobile application aimed at improving pediatric dental hygiene through the integration of gamification strategies. The application, designed for the Android platform, leverages the React Native framework to ensure cross-platform compatibility, focusing on engaging children in effective oral hygiene practices with minimal parental supervision. The gamification approach integrates a main game character that guides the user through the tooth brushing process, making the routine both fun and educational. The application also features a secure, PIN-protected parental interface that allows for the monitoring and tracking of children's dental hygiene habits. This study involved the iterative design, implementation, and testing of the application, followed by an evaluation phase where usability was assessed through the System Usability Scale (SUS) and behavioral changes were monitored. The results indicate that gamification can significantly enhance children's engagement and consistency in dental hygiene practices. This research not only confirms the feasibility of using mobile technology and gamification in health education but also provides a scalable model that could be expanded to other platforms and health behaviors.

# Acknowledgements

I would like to thank Dr. Bruce Scharlau, my project supervisor, who has provided me with his invaluable guidance, support and time. Throughout this project and my whole time at university, Dr. Scharlau has been an incredible role model and has shown me an abundance of patience.

I would also like to thank Plamena Mya, the person who conceptualised this application and believed that I was the person who could bring her vision to life.

Lastly, I would like to thank the University of Aberdeen's Computing Science staff who helped me grow academically.

.

# Contents

0.1	Abbreviations . . . . .	11
<b>1</b>	<b>Introduction</b>	<b>12</b>
1.1	Background . . . . .	12
1.2	Project Motivation . . . . .	12
1.3	Aims and Objectives . . . . .	12
1.3.1	Primary Aims . . . . .	12
1.3.2	Secondary Aims . . . . .	13
1.4	Document Structure . . . . .	13
<b>2</b>	<b>Background</b>	<b>15</b>
2.1	Pediatrics and Parental Influence . . . . .	15
2.1.1	The United Kingdom . . . . .	15
2.2	Adequate techniques for dental hygiene . . . . .	16
2.3	Overview of Technological Engagement Among Children . . . . .	17
2.4	Competitive Landscape . . . . .	17
2.4.1	Android and iOS . . . . .	18
2.4.2	Alternative digital platforms . . . . .	21
2.4.3	Comprehensive Insights from Competitive Landscape Evaluation . . . . .	22
2.5	Gamification . . . . .	22
2.5.1	Foundations of Gamification . . . . .	22
2.5.2	Historical Context and Evolution . . . . .	23
2.5.3	Reward Systems in Gamification . . . . .	24
2.5.4	Utilising a Game Character and The Tetrad Framework . . . . .	25
2.5.5	Effective Educational Applications . . . . .	27
2.5.6	Challenges and Ethical Considerations . . . . .	27
<b>3</b>	<b>Technical Review and System Architecture</b>	<b>28</b>
3.1	Introduction to the System Architecture . . . . .	28
3.2	Backend . . . . .	29
3.2.1	Framework . . . . .	29
3.2.2	Django and RESTful APIs . . . . .	29
3.2.3	Database Management . . . . .	32
3.2.4	Backend Deployment . . . . .	33
3.2.5	Environment Variables . . . . .	33

3.3	Frontend . . . . .	33
3.3.1	Initial Approach . . . . .	34
3.3.2	Final Approach . . . . .	34
3.4	Backend Architecture . . . . .	35
3.5	Frontend Architecture . . . . .	37
3.6	Frontend-Backend Relationship . . . . .	39
3.6.1	Data Exchange and API Communication . . . . .	39
3.6.2	Practical Example . . . . .	39
3.7	Implementation . . . . .	40
3.8	Software Engineering Practices . . . . .	40
3.8.1	Scrum Adaptation . . . . .	40
3.8.2	Lean Adaptation . . . . .	41
3.8.3	Kanban Adaptation . . . . .	41
3.8.4	Extreme Programming (XP) Adaptation . . . . .	42
3.9	Risk Assessment and Project Timeline . . . . .	43
3.9.1	Risk Assessment . . . . .	43
3.9.2	Project Timeline and Development Cycles . . . . .	46
<b>4</b>	<b>Implementation</b>	<b>47</b>
4.1	Development Environment Setup . . . . .	47
4.1.1	Integrated Development Environment (IDE) and Code Editors . . . . .	47
4.1.2	Graphics and Diagram Creation . . . . .	47
4.1.3	Database Management Software . . . . .	48
4.1.4	Platforms and Testing Devices . . . . .	48
4.1.5	Version Control . . . . .	48
4.2	Dependency Management . . . . .	49
4.2.1	Frontend Dependencies . . . . .	49
4.2.2	Backend Dependencies . . . . .	51
4.3	App workflow and Wireframes . . . . .	53
4.3.1	Log In, Sign Up and Home screens . . . . .	53
4.3.2	Home, Progress and Shop Screens . . . . .	54
4.3.3	Parents, Statistics and Settings Screens . . . . .	56
4.4	Communication Between Components: System Architecture Diagram . . . . .	56
4.5	Design Choices . . . . .	57
4.5.1	Naming the Application . . . . .	57
4.5.2	Graphics and Demographic Considerations . . . . .	57
<b>5</b>	<b>Evaluation</b>	<b>59</b>
5.1	Introduction . . . . .	59
5.2	Methodology . . . . .	59
5.3	Statistical Methods Employed . . . . .	59
5.4	Results . . . . .	60
5.4.1	SUS Results . . . . .	60

5.4.2	Descriptive Statistics . . . . .	60
5.4.3	Frequency Distribution: . . . . .	60
5.4.4	<b>Behavioral Changes</b> . . . . .	61
5.5	Conclusions . . . . .	61
<b>6</b>	<b>Results</b>	<b>62</b>
6.1	SUS Results . . . . .	62
6.2	Descriptive Statistics . . . . .	62
6.3	Frequency Distribution . . . . .	62
6.4	Behavioral Changes . . . . .	62
6.5	Limitations of the Study . . . . .	62
6.5.1	Selection Bias . . . . .	63
6.5.2	Self-reported Data . . . . .	63
6.5.3	Lack of Diverse Demographic Data . . . . .	63
<b>7</b>	<b>Discussion</b>	<b>64</b>
7.1	Project Summary . . . . .	64
7.2	What worked well . . . . .	64
7.2.1	For The Author . . . . .	64
7.2.2	Application-wise . . . . .	65
7.3	Challenges . . . . .	65
7.4	Future Implementations . . . . .	65
7.4.1	Interactive Character Development . . . . .	65
7.4.2	Verification of Brushing Activity . . . . .	65
7.4.3	Design and Accessibility Enhancements . . . . .	66
7.4.4	Parental Educational Tools . . . . .	66
7.4.5	User Engagement and Security Features . . . . .	66
7.5	What I Would Do Differently If I Had To Start All Over Again And What I Have Learnt During This Project . . . . .	67
	<b>Appendices</b>	<b>68</b>
<b>A</b>	<b>User Evaluation Survey</b>	<b>69</b>
<b>B</b>	<b>Survey Responses</b>	<b>73</b>
<b>C</b>	<b>User Manual</b>	<b>84</b>
C.1	Introduction . . . . .	84
C.2	System Requirements . . . . .	84
C.3	Installation Instructions . . . . .	84
C.3.1	Setup Android Studio . . . . .	84
C.3.2	Download the Application . . . . .	84
C.3.3	Install Dependencies . . . . .	84
C.3.4	Start the Application . . . . .	85

---

C.4	Operating Instructions . . . . .	85
C.5	Troubleshooting . . . . .	85
C.5.1	Common Issues . . . . .	85
<b>D</b>	<b>Maintenance Manual</b>	<b>86</b>
D.1	Introduction . . . . .	86
D.2	System Overview . . . . .	86
D.3	Software Dependencies . . . . .	86
D.3.1	Frontend . . . . .	86
D.3.2	Backend . . . . .	86
D.4	Installation and Setup . . . . .	86
D.4.1	Frontend Setup . . . . .	86
D.4.2	Backend Setup . . . . .	86
D.5	File Structure . . . . .	87
D.5.1	Frontend . . . . .	87
D.5.2	Backend . . . . .	87
D.6	Making Changes . . . . .	87
D.6.1	Frontend Changes . . . . .	87
D.6.2	Backend Changes . . . . .	88
D.7	Bug Reporting and Monitoring . . . . .	88



## List of Tables

3.1	User Model . . . . .	36
3.2	Connection between Views and URLs in the application . . . . .	37
3.3	React Native Frontend Architecture Components . . . . .	39
3.4	Extensive Risk Assessment for Application Development . . . . .	45

# List of Figures

2.1	Step-by-step guide for brushing teeth with a manual toothbrush . . . . .	17
2.2	Search interest of the term gamification since January 2010 . . . . .	23
2.3	The Flow Channel . . . . .	24
2.4	The Lens of Endogenous Value . . . . .	25
2.5	The Tetrad Framework . . . . .	26
3.1	Simplified version of the frontend-backend relationship . . . . .	28
3.2	Example of data conversion from JSON to Python using Django REST Framework . . .	30
3.3	Pseudocode showing a Django REST Class-Based View for deleting a user . . . . .	31
3.4	Browsable API page, configured only for <i>POST</i> requests. . . . .	31
3.5	Browsable API Interface . . . . .	32
3.6	Example of what an Environment <i>.env</i> file typically looks like . . . . .	33
3.7	Client-server interaction within the Django REST Framework . . . . .	35
3.8	Yoga Library . . . . .	38
3.9	Client-server interaction within Django REST - Detailed . . . . .	40
3.10	Kanban Board Visualization . . . . .	42
3.11	Pseudocode for testing a successful signup functionality in a Django application . . . .	43
3.12	Pseudocode for testing successful component rendering in React Native . . . . .	43
3.13	Project Gantt Chart . . . . .	46
4.1	Wireframe for the Log In, Sign Up and Home screens . . . . .	53
4.2	Wireframe for the Home, Progress and Shop screens . . . . .	54
4.3	Wireframe for the Parents, Statistics and Settings screens . . . . .	56
4.4	Wireframe for the Home, Progress and Shop screens . . . . .	58
5.1	User Evaluation question about the need for parental supervision . . . . .	60

## 0.1 Abbreviations

<b>API</b>	Application Programming Interface
<b>CLI</b>	Command Line Interface
<b>CORS</b>	Cross-Origin Resource Sharing
<b>CRUD</b>	Create, Read, Update, Delete
<b>CSRF</b>	Cross-Site Request Forgery
<b>DRF</b>	Django Rest Framework
<b>EAS</b>	Expo Application Services
<b>HHTP</b>	Hypertext Transfer Protocol
<b>HTTPS</b>	Hypertext Transfer Protocol Secure
<b>IDE</b>	Integrated Development Environment
<b>JSON</b>	JavaScript Object Notation
<b>JWT</b>	JSON Web Token
<b>MVC</b>	Model-View-Controller
<b>MVS</b>	Model-View-Serializer
<b>OCC</b>	Optimistic Concurrency Control
<b>ORM</b>	Object-Relational Mapping
<b>OS</b>	Operating System
<b>PaaS</b>	Platform As a Service
<b>RDBMS</b>	Relational Database Management System
<b>REST</b>	Representational State Transfer
<b>SUS</b>	(Needs a definition to place alphabetically)
<b>UI</b>	User Interface
<b>UX</b>	User Experience
<b>VS Code</b>	Visual Studio Code
<b>XP</b>	Extreme Programming
<b>XP</b>	Experience Points
<b>XSS</b>	Cross-Site Scripting

## Chapter 1

# Introduction

### 1.1 Background

Oral hygiene is pivotal to people's well-being (Đorđević, 2018). Nevertheless, familiarity and adherence to proper oral hygiene habits, particularly about the adequate brushing techniques, is often overlooked (Đorđević, 2018; Ganss et al., 2008). The acquisition of knowledge about the effective practices happens in the early stages of childhood when habit formation begins (Alm et al., 2007). However, a surprising number of children lack knowledge of these practices (Public Health England, 2022; Özbek et al., 2017; Green et al., 2023), which underscores a huge health concern as poor oral hygiene can lead to numerous health issues and impact one's overall wellness (Dirks and Monopoli, 2019).

### 1.2 Project Motivation

This project is conceptualised by Plamena Mya, a dental hygienist, who has observed the consequences of inadequate oral hygiene habits firsthand through her job. Additionally, as part of the National Health Service<sup>1</sup> (NHS), Plamena has seen the exceptionally high demand dentists experience daily. Her belief is that through educating children early on about the proper brushing techniques, the pressure on dentists would be reduced, as adhering to these practices would result in healthier teeth (Abijeth. and A.C, 2017). After a comprehensive discussion with the author of this paper, and research on the current market, a decision to leverage mobile technologies to create an innovative application utilising gamification strategies that would serve as an educational tool for children was taken. This application would be designed for a young audience and available initially across the United Kingdom.

### 1.3 Aims and Objectives

To ensure the application's accessibility across different platforms and the potential of scaling up significantly, the student employed the React Native<sup>2</sup> and Django Representational State Transfer<sup>3</sup> (REST) Frameworks. As mentioned above, the main goal of the project is to develop an application that serves as an educational tool. To ensure the timely delivery of a finished project, the author split the aims and objectives into primary and secondary as outlined in the two subsections below.

#### 1.3.1 Primary Aims

The following goals were put forward as the main aims for the application to achieve:

---

<sup>1</sup><https://www.nhs.uk/>

<sup>2</sup><https://reactnative.dev/docs/environment-setup>

<sup>3</sup><https://www.django-rest-framework.org/>

- **Brushing Teeth Functionality:** A functionality that would allow children to learn and practice adequate brushing techniques with minimal parental supervision.
- **Game Character:** Utilising gamification techniques, the application would feature a main character to guide children through the brushing process, making the learning process more enjoyable and engaging.
- **Parental Interface:** A PIN-protected parental interface that would enable parents to track their child's brushing habits, and view achievements, such as consecutive days of brushing, or if their child has the tendency to brush their teeth more often in the evening/morning.
- **Levels:** Based on how actively users engage with the application, there would be levels that can be progressed through.
- **Data Security:** Given the potential future scaling up of the application that would require for managing large amounts of sensitive data, ensuring that data is securely stored had to be listed as one of the main objectives to guarantee the application's reliability.

### 1.3.2 Secondary Aims

The following goals were listed to be implemented if the primary goals were met within the timeframe of the project:

- **Mini Shop:** This functionality would be implemented to allow children to personalise the look of the game character, enhancing user engagement and retention.
- **Progress Bar:** To visualise the progress through a given level, allowing children to track their advancement in real time.
- **Social Features:** Integrating social functionalities like adding friends and starting joint brushing sessions would encourage interaction and would build a community around healthy habits, aligned with the social development needs of the target age group.
- **iOS implementation:** Given React Native's cross platform nature, most of the maintained code-base would be identical for both Android and iOS. However, some components differ and therefore the author decided to focus on Android, leaving iOS' implementation of the application as a secondary aim.

## 1.4 Document Structure

The remainder of this paper is structured to guide the reader through the following chapters, relevant to the chronological order of the project's timeline:

- **Chapter 2: Background** - Discusses the foundation of pediatric dental hygiene, the role of parental influence, and introduces the concept of gamification in health education.
- **Chapter 3: Technical Review and System Architecture** - Details the technical specifications of the system, including the architectural design, choice of technologies, and system components.
- **Chapter 4: Implementation** - Covers the detailed process of developing the application, from the setup of development environments to the deployment and testing phases.

- 
- **Chapter 5: Evaluation** - Presents the methodologies and results of evaluating the application through user testing and statistical analysis.
  - **Chapter 6: Results** - Provides a detailed analysis of the data collected from the application usage and its impact on the target demographic.
  - **Chapter 7: Discussion** - Offers insights into the implications of the findings, discusses the project's limitations, and suggests areas for future research.
  - **Appendices:** - Includes additional materials such as the User Manual, Maintenance Manual, survey questionnaires, and raw data tables that support the thesis.

## Chapter 2

# Background

### 2.1 Pediatrics and Parental Influence

As mentioned in the introduction 1, the early years of childhood are pivotal for individuals to acquire habits that will impact their health. Among these habits, a critical one is oral hygiene (Đorđević, 2018). Nevertheless, adequate dental hygiene practices among children are notably low. Research shows that there is a substantial gap in young individuals' knowledge about the proper brushing techniques, with a great number of children not adhering to the recommended practices or not doing so consistently (Public Health England, 2022; Özbek et al., 2017).

This aforementioned deficiency in oral health knowledge is often rooted in the generational transfer of habits, where the parents serve as the primary role models to teach those. A questionnaire by Özbek et al. (2017) <sup>1</sup> found that approximately 92.1% of the children had learnt brushing techniques from their parents/guardians. Unfortunately, research by Đorđević (2018) <sup>2</sup> shows that parents often do not know the proper brushing technique themselves or do not apply it (Ganss et al., 2008). Or alternatively, they do not have the time to supervise their children every time they brush their teeth or lack the motivation to do so.

Additionally, there is a lack of educational content in schools and digital platforms that is both engaging and fun and could serve as the solution to this pressing concern. Further supporting this claim, in their findings, Özbek et al. identified that factors contributing to irregular brushing habits, as reported by parents, were lack of motivation (23%), forgetfulness (15.9%), and a lack of educational oral hygiene materials (18.3%).

However, as noted by Kiatipi et al.<sup>3</sup>, negligence of oral hygiene habits could be directly linked to a spectrum of adverse health outcomes such as dental decay, gum disease, and oral infections which could pose long-term health risks and significantly impact the quality of an individual's life. These conditions not only affect oral health but are also directly linked to an increased risk of other diseases, which further underscores the integral role of oral health (Dirks and Monopoli, 2019). In the following subsection, the author will discuss the current state of pediatric dentistry in the United Kingdom.

#### 2.1.1 The United Kingdom

The landscape of pediatric dentistry within the United Kingdom (the UK) is marked by challenges within the National Health Service (NHS) in ensuring adequate access to dental care and promoting effective oral hygiene practices.

---

<sup>1</sup>(Özbek et al., 2017)

<sup>2</sup>(Đorđević, 2018)

<sup>3</sup>(Kiatipi et al., 2021)

A comprehensive survey conducted by Public Health England (2022)<sup>4</sup> provides a pivotal benchmark in understanding the current oral health state among young children. The findings of the National Dental Health Epidemiology Programme (NDEP) revealed that in 2022, approximately 23.4% of 5-year-old children experienced dental decay, indicating a pressing dental health issue. Furthermore, data from the 2013 Children's Dental Health (CDH) Survey<sup>5</sup>, which assesses the dental health of the nation's children every 10 years, indicated that approximately a third of 5-year-olds and nearly a half of 8-year-olds had suffered dental decay in their primary teeth.

Compounding these issues, a BBC News report (Green et al., 2023) underscores the serious pressures faced by the NHS dental services. As mentioned by the article, wait lists for pediatric dental assistance are "exceptionally high", with more than 12 000 underage individuals awaiting services at the start of 2023. The most prolonged wait recorded for a patient stretched to 80 weeks. The article further reveals these challenges, mentioning a visit by dentists from a charity team to primary schools across Hull and parts of East Yorkshire. In a single day, among 169 children examined, dentists discovered 263 decayed teeth, illustrating the dire need for improved oral hygiene practices among young children. Proper oral hygiene maintenance and adherence to adequate brushing techniques could potentially reduce this frequency of dental decay among children and consequently, this would likely alleviate some of the demand on the NHS dental services (Abijeth. and A.C, 2017).

In the next section, the author will briefly discuss the proper brushing techniques that could help in the reduction of cavities.

## 2.2 Adequate techniques for dental hygiene

Adequate teeth brushing could be achieved with both traditional manual or electric toothbrush. Regardless of type of toothbrush used, the key to effective cleaning lies in the technique employed by individuals. The NHS Inform<sup>6</sup> guide and insights from Plamena Mya (who is an experienced dental hygienist) provided an essential direction. The proper technique involves placing the toothbrush at a 45-degree angle to the gums, followed by gentle back-and-forth motions in short strokes, or in small circular movements, ensuring that all tooth surfaces are brushed. It's recommended to spend at least 2 minutes following this technique, focusing on each quadrant of the mouth. Additionally, individuals should remember to brush the tongue to remove bacteria. This comprehensive approach is depicted in figure 2.1 (Oceana Dental, 2021).

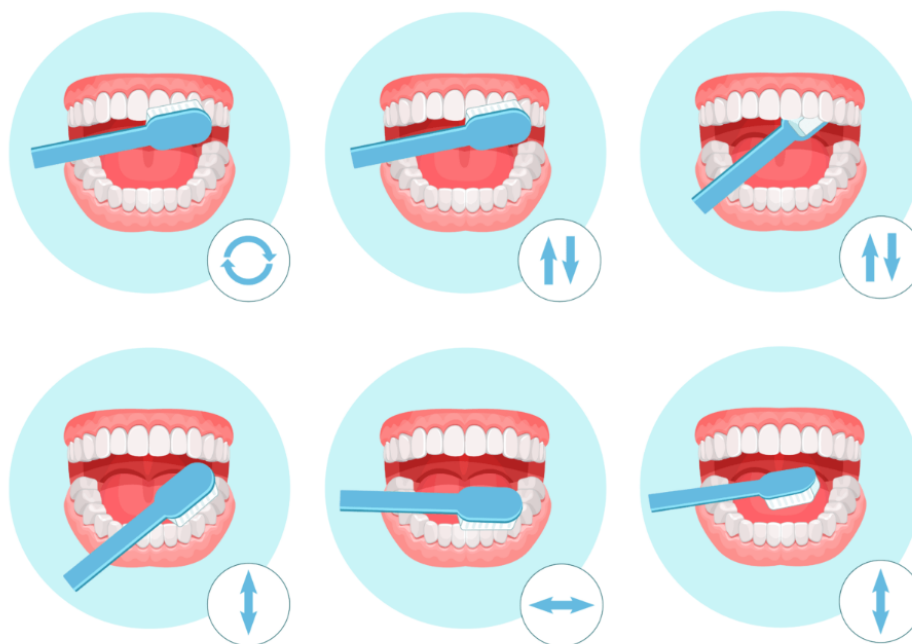
---

<sup>4</sup>(Public Health England, 2022)

<sup>5</sup>(Children's Dental Health (CDH) Survey, 2013)

<sup>6</sup>(NHS Inform, 2024)





**Figure 2.1:** Step-by-step guide for brushing teeth with a manual toothbrush (Oceana Dental, 2021). The proper technique involves placing the toothbrush at a 45-degree angle to the gums, followed by gentle back-and-forth motions in short strokes, or in small circular movements, ensuring that all tooth surfaces are brushed.

## 2.3 Overview of Technological Engagement Among Children

In recent years, there has been a remarkable increase in the usage of digital devices among children, as indicated by various studies (Chang et al., 2018) and statistics illustrating the rapid growth of technology shown by substantial increase in the amount of time children spend on digital devices, including smartphones, tablets, computers, and gaming consoles (L., 2023). The author asserts that this trend could be utilised by leveraging mobile devices and tablets to offer an innovative solution to the increasing problem in children's dental hygiene mentioned in Subsection 2.1.1 by creating an application that could potentially enhance oral hygiene techniques among younger individuals.

In the next couple of subsection the author will evaluate the current competitive landscape for such application to assess whether there is a critical need for an innovative application.

## 2.4 Competitive Landscape

Indeed, the market is saturated with a plethora of applications, available on both mobile and tablet devices, as well as alternative digital platforms. To identify whether there is a necessity for a novel application focusing on the same domain, the author undertook a comprehensive evaluation of the current competitive landscape to assess it. The selection criteria for these digital solutions were based on quantitative metrics, including download numbers, user ratings, and, for content-based platforms, view counts (Mackey et al., 2022). The evaluation will begin by a discussion of the available applications for iOS and Android.

### 2.4.1 Android and iOS

This subsection is dedicated to an in-depth evaluation the applications of dental hygiene applications available on the two leading mobile operating systems: iOS and Android. While the exact details of the search algorithms for both the App Store <sup>7</sup> (iOS) and Google Play <sup>8</sup> (Android) have not been fully disclosed, the number of downloads, number of downloads in a recent time frame, user engagement, behaviour, and retention seem to be factors influencing search results. The following subsections describe applications relative to the "popularity" arrangement for both platforms. This comparison aims not only to identify the most popular applications but also to understand the factors contributing to their success and user adoption as well as recognise flaws in they system design and issues users might come across. To achieve this, the author downloaded the applications on their personal devices and tested them thoroughly.

#### Magic Timer

##### Features and Observations

- **Parent Mode:** Includes statistical tracking with a simple PIN code system. The single-digit code could be easily bypassed by children, questioning the effectiveness of this security measure.
- **Report Card:** In the parent section, users can find a log containing information about their children's current streak , total stars earned and how many times their child has brushed their teeth over the last 30 days.
- **Timer Functionality:** The main functionality of the application employs a *Mickey Mouse* timer with a blurred screen that progressively clears during a 2-minute brushing session. The screen is divided into four quadrants, each lasting 30-seconds. However it was that was inordinately challenging for the author to comprehend, and possibly near impossible for a child grasp as the application did not provide any instructions or cues about its use. Additionally, the brushing functionality does not allow for a user to pause a current session, limiting user experience. Furthermore, there are no provided guidelines to inform individuals of the adequate brushing techniques.
- **Sound Loops:** Features a 10-second sound loop intended to engage users during brushing, though the short loop may become repetitive.
- **Reward System** Upon completion of a brushing session, users earn *Mickey Mouse* stickers and badges-either half or a full badge based on whether the user completed 1 or 2 brushing sessions during that day.
- **Home Screen Design:** Displays a map with buttons resembling level indicators that could be clicked on. However, in reality it is a calendar presenting the days of the current month in an ambiguous fashion.
- **Customisation:** Allows character and toothbrush color customization, though additional options require in-app purchases, limiting access to varied content.

---

<sup>7</sup>(App Store, 2024)

<sup>8</sup>(Google Play, 2024)

### Areas for improvement

- **Educational Content:** While the application utilises engaging *Disney characters*, it misses the crucial opportunity to educate young individuals on adequate brushing techniques or oral hygiene practices.
- **Gamification and Engagement:** An attempt to use gamification for enhancing user engagement is present. However displaying the character during the brushing timer and diversifying rewards could significantly improve user engagement and overall experience.
- **Design:** Additionally, employing a simplified design that is readily comprehensible to children could benefit the application's performance significantly and potentially increase its popularity.

### Brushout

#### Features and Observations

- **Interactive design:** The main brushing functionality of the application employs a 2-minute timer with a 2-D sketch of human teeth. This visual aid highlights which mouth quadrant should be brushed. While visually instructive, the lack of engaging auditory elements beyond a simple ding sound might limit its appeal.
- **Voice Coach:** Additionally, the application offers basic prompts during brushing (e.g. "Let's go", "Switch sides", "Next part"), but their simplicity might not significantly enhance the brushing experience and does not provide details of the adequate brushing technique.
- **Premium Options:** The application offers a premium subscription at various price points (monthly, annually, lifetime), which includes features like a manual journal, custom routine, and more settings. However, there is no further clarification within the application, informing what those features entail and may hinder user interest in upgrading their subscription.
- **Calendar Tracking:** Features a calendar screen which displays total number of brushing seconds per day. However, after the author tested the brushing functionality a couple of times, the calendar did not update, which could make users question its reliability.
- **Educational Content:** The application provides how-to-guides that redirect users to YouTube videos. Additionally, on the same screen, users' current and longest streak are displayed. This approach misses the opportunity for integrated, direct educational content within the application. Moreover, it would be more appropriate to display the streak information on the calendar screen.
- **Rewards System:** The application provides an achievements tab, which is somewhat obscured within the Information screen and might be ambiguous for users to find. This tab offers a total of four achievements. The milestones include the first brush, 1-week streak, 1-month streak, and reaching 100 brushes, which may not effectively motivate long-term user engagement.

### Areas for improvement

- **User Engagement:** Users could potentially be more effectively retained within the application if a more interactive and motivational features were employed to capture their interest.

- **Calendar Functionality:** Addressing the technical issues within the calendar and possibly other areas of the application would improve its utility as a tool to monitor brushing habits.
- **Integrated Educational Content:** Incorporating the educational content linked from YouTube directly within the application could provide users with a more seamless learning experience.
- **Rewards System:** A diversified rewards system with more achievements and visible incentives could motivate users to maintain and improve their oral health.

## OralB app

### Features and Observations

- **User Experience (UX) Design:** Offers similar, superior, design to the *Brushout* application. Additionally, the application employs a user creation functionality, which would allow individuals to save their data and access it from other devices.
- **Electronic Toothbrush:** The application works with *OralB's* electronic toothbrush, which could potentially deter users from downloading the application as they would have to purchase a specific toothbrush. Moreover, some individuals prefer using a traditional manual toothbrush.
- **Main Functionality:** According to the application's website, the brushing functionality is supposed to provide users with personalised guidance, based on data from users' everyday oral care routine. Furthermore, users receive real-time feedback on a 3-D map from *OralB's* built-in AI Brushing Recognition Technology.
- **Reward System:** The application offers just 12 analogous rewards, which might not effectively motivate long-term user engagement.

### Areas for Improvement

- **Child-Friendly Design:** The application could derive substantial benefits from the incorporation of child-friendly design elements. Although the system offers some educational content and real-time feedback, it is clear from the screen captures that those have been made to cater to adults and not children.
- **Reward System:** Diversified rewards system would engage users and encourage adhering to adequate brushing techniques.

## Brush Monster

### Features and Observations

- **UX Design:** Simplified, child-friendly design that would be favourably seen by children. Additionally, users can choose between 3 animal characters that guide them through the application.
- **Free Trial:** Although the application is paid, a 7-day free trial is offered to users.
- **Facial Recognition:** The main functionality of the application utilises facial recognition which based on a user's face shows approximately which mouth quadrant should be brushed, followed by simple instructions providing information about brushing technique.

- **Voice Coach:** The application employs a child-like voice for the brushing instructions and in a combination with the facial recognition algorithm pauses if the user is not present in front of the camera and prompts the user to come back and resume their session (e.g. "Where are you? I can't see your face.").
- **Reward System:** Stickers which can be collected by completing brushing sessions.
- **Parent Mode:** Similar to *Magic Timer*, *Brush Monster* employs a PIN protected parent mode. However, *Brush Monster's* security measure is to ask for the result to a simple mathematical problem, which ensures the actual protection of this screen against young children. This screen provides information about the electronic toothbrush supported by the application (e.g. how many times it has been used or when to change it) if users opt for it. Additionally, the parent mode provides information about the child's current streak to track their progress.

### Areas for Improvement

- **Educational Content:** The application misses the opportunity for more direct educational content within the main functionality. Additionally, providing simple cues about which mouth quadrant should be brushed and how would be highly beneficial for children to learn and adhere to proper brushing techniques.
- **Reward System:** The application could potentially benefit from a diversified reward system, which may motivate long-term user engagement.
- **Parent Mode:** More statistical information about user's progress and application use would be beneficial for parents to track their child's progress.

### Alternative Applications

The author additionally came across some alternative applications which might be utilised by parents during their child's brushing time. However, most of them do not employ gamification elements and are simple timers, or for the case that they offer a brushing functionality, the user is required to purchase the application's electronic toothbrush to even access them (e.g. *GUM Playbrush*).

#### 2.4.2 Alternative digital platforms

This subsection will explore the competitive landscape that the author came across which was not available on Google Play and the App Store, but on the Internet. Listed Below are the top 3 most viewed and liked children songs on YouTube about tooth brushing.

- Brush Your Teeth | Kids Songs | Super Simple Songs
- Brush Your Teeth | Kids Songs | Finny The Shark
- Brush Your Teeth | Early Years - Nursery Rhymes by BBC

A vast majority of those songs employ catchy, child-likeable songs to walk viewers through the 2-minute brushing process, however, they fail to successfully demonstrate adequate brushing techniques. Although, this could be due to the fact it would be exceptionally difficult to create a video that would

both be educational and fun to watch. Additionally, in Scotland, the Childsmile Toothbrushing Programme<sup>9</sup> has grown huge popularity. Childsmile is a supervised Toothbrushing Programme that adheres to national standards and provides children with toothbrushes, toothpastes and guidance on brushing techniques at nurseries and schools. It is an excellent programme promoting oral health. The only limitation of Childsmile is that its way of making an impact on children is through in-person visitations which may not be enough for children to form healthy oral health habits .

### 2.4.3 Comprehensive Insights from Competitive Landscape Evaluation

From all of the research the author conducted, they observed a few things:

- **Educational Content:** Applications fail to impart knowledge about adequate brushing techniques to a young audience.
- **Gamification Element:** Applications fail to incorporate actual gamification elements that would make them more engaging and keep young users longer.
- **Technical Part:** None of the applications examined showed a complex structure with different functionalities supported. Moreover, the ones that did offer such, failed to do so in a correct manner.
- **Rewards System:** Applications fail to incorporate meaningful and engaging reward system, that would serve as a motivational tool for children. The author will examine what exactly is meant by "meaningful" reward systems in Section 2.5.3.
- **Alternative Approaches:** Although there are alternative approaches available, they require parental supervision and guidance, or are only available in schools/nurseries and not at home.

Considering all of the above and the critical oral health situation, the author reached the conclusion that there is a critical need for an innovative application that will teach children how to adhere to adequate brushing techniques, done in an entertaining manner, utilising gamification elements. In the next section, the author will examine what exactly gamification is.

## 2.5 Gamification

.

### 2.5.1 Foundations of Gamification

For centuries games have provided means of entertainment for individuals, especially children, in diverse forms such as physical activities, board games, and card games (Levine, 2008). Recently, businesses have begun making profit and popularising their applications via the medium of games to make them more appealing and successful to their customers (Wen et al., 2014). This strategy is also known as gamification<sup>10</sup> and involves the utilisation of game-design features and principles into a non-game context. The primary aim of gamification is to increase user engagement and increase the likelihood of user retention. Typical elements of gamification include, but are not limited to, badges, challenges, levels and leaderboards. In the next subsection the author will discuss how has gamification evolved throughout the years and how the implementation of these strategies has shifted with advances in technology.

---

<sup>9</sup><https://www.childsmile.nhs.scot/>

<sup>10</sup><https://dictionary.cambridge.org/dictionary/english/gamification>

### 2.5.2 Historical Context and Evolution

Gamification is a relatively recent notion that has just become popular in the 21st century. In fact, one of the first recorded applications utilising gamification is the MUD1 project, developed by Richard Bartle in 1980 (Tezcan and Richards, 2011). MUD1<sup>11</sup> was a text-based system that allowed users to enter a shared virtual world, the first of that kind. Although it did not fully encapsulate what we now recognise as gamification, MUD1 has contributed to the foundational concepts that have influenced the broader genre of multiplayer gaming (Dieterle, 2007) such as in *World of Warcraft*<sup>12</sup> and *Fortnite*<sup>13</sup>. Gamification as we know it now has evolved via two primary movements.

The first one, exemplified by Tom Malone's research in MIT, focused on understanding what makes games "fun" and engaging. In his paper *Toward a Theory of Intrinsically Motivating Instruction* (Malone, 1981), professor Malone explores how game design elements such as challenge, fantasy and curiosity can make an activity enjoyable and engage players more deeply.

Concurrently, the "Serious Games" movement (Ahrens, 2015), started by Ben Sawyer and David Reetsky in 2002 focused more explicitly on the educational perspective games can bring. This movement looked at different ways to utilise games beyond entertainment to serve for educational purposes by employing their engaging nature. The term "gamification," as it is currently understood, was initially used in 2003 by Nick Pelling while designing a game-like user interface for ATMs (Frost et al., 2015). Since then, Pelling's creative method has spawned a plethora of programs that improve commonplace processes and activities by including game-like features like engagement strategies and prizes.

To look into gamification's current popularity, the author examined *Google Trends*<sup>14</sup>. According to data presented in Figure 2.2 there has been a remarkable increase in the searches for the term *gamification*, reaching its peak in April 2023 with 100 searches. The term "gamification learning" emerged as the second most related query, highlighting the growing interest in applying gamification strategies to educational contexts. In the next two subsections, the author will speak about meaningful reward systems and utilising game characters in the context of educational application for which they drew inspiration by the *A Theory of Fun for Game Design* and *Art of Game Design: A Book of Lenses* books by Raph Koster and Jesse Schell respectively.



**Figure 2.2:** Search interest of the word *gamification* since January 2010 according to Google Trends<sup>15</sup>. There has been a remarkable increase in the searches for the term, reaching its peak in April 2023 with 100 searches.

In the next sections, the author will detail some of the most prominent gamification elements according

<sup>11</sup><https://en.wikipedia.org/wiki/MUD1>

<sup>12</sup><https://worldofwarcraft.blizzard.com/en-gb/>

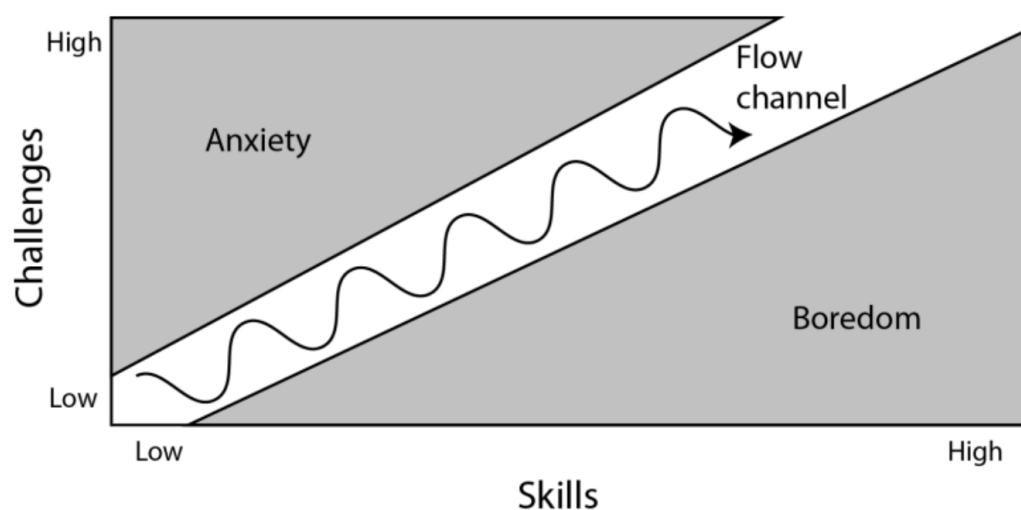
<sup>13</sup><https://www.fortnite.com/>

<sup>14</sup><https://trends.google.com/trends/>

to literature.

### 2.5.3 Reward Systems in Gamification

Since reward systems are the primary source of long-term user engagement and motivation, gamification heavily depends on them. In his book *A Theory of Fun for Game Design* (Koster, 2014, Ch.5, p. 96), Raph Koster suggests that fun "is the feedback the brain gives us when we are absorbing patterns for learning purposes". This idea holds that people see learning as a form of entertainment, and rewards act as the feedback mechanism to reinforce this idea. Koster also provides an example of two basketball teams. The first team, played a game to "have fun", and the second team played to win. He explains how the latter is no longer approaching the game as practice and implies that consistency, in combination with enjoyment, is more conducive to habit formation than striving for mastery. Additionally, Koster suggests that developers should deliver to consumers engaging new features that are not exceedingly strenuous. This is a result of the need for applications to allow users to progress in a way that is consistent with their present skill level without becoming monotonous. Jesse Schell (2008) provides more support for this claim in his book *Art of Game Design: A Book of Lenses*. Schell states that for players to acquire the expected knowledge or skills from a game, they have to be kept in *The Flow Channel* (Koster, 2014, Ch. 5) which is often associated with Mihaly Csikszentmihalyi's work on flow theory (Abuhamdeh, 2020). Figure 2.3 provides a visual representation of this state-finding the right balance between challenges and skills, pivotal for educational games. To keep users entertained and able to meet the learning expectations, they need to be in the *flow*.



**Figure 2.3:** *The Flow Channel*- a mental state that helps us maintain attention on a task by finding the right balance between challenges and skills as defined by Koster (2014).

This is where rewards and their quality become pivotal. Koster explains how "if an activity does not give a quantifiable reward, they'll think it's dull" (Koster, 2014, Ch.7, p.121). Similarly, the *Lens of Endogenous Value* (Jesse Schell, 2008, Ch.3, p33), as illustrated in Figure 2.4, is introduced in the *Art of Game Design: A Book of Lenses* book, which in essence implies that rewards must have a meaning to players and feel valuable. The author of the book further explains this by giving an example with the game *Bubsy*<sup>16</sup> for SNES and Sega Genesis. This game utilises a cat character, which has to be navigated

<sup>16</sup><https://en.wikipedia.org/wiki/Bubsy>



### *Lens #5: The Lens of Endogenous Value*

To use this lens, think about your players' feelings about items, objects, and scoring in your game. Ask yourself these questions:

- What is valuable to the players in my game?
- How can I make it more valuable to them?
- What is the relationship between value in the game and the player's motivations?

Remember, the value of the items and score in the game is a direct reflection of how much players care about succeeding in your game. By thinking about what the players really care about and why, you can often get insights about how your game can improve.

**Figure 2.4:** *The Lens of Endogenous Value* which implies that rewards must have a meaning to players and feel valuable (Jesse Schell, 2008, Ch.3, p33)

through levels while defeating enemies and collecting points. However, it can be quickly observed that the points collected throughout the game serve no actual purpose and cannot be exchanged for anything. Schell states that players would initially collect points with the expectation of them being valuable, however, they would quickly give up on it and focus just on finishing the game. This is due to the fact that winning the game is the players' intrinsic motivation and earning points does not align with this goal.

Additionally, to focus on the learning process and the characteristics that define a learning experience in a game, the author was inspired by Koster (Koster, 2014, Ch7, p.121). Koster states that "High-level players can't get big benefits from easy encounters or they will bottom-feed. Inexpert players will be unable to get the most out of the game". This assertion implies precisely what Schell does: games should be challenging relative to the player's experience in order to stay in the *Flow Channel*.

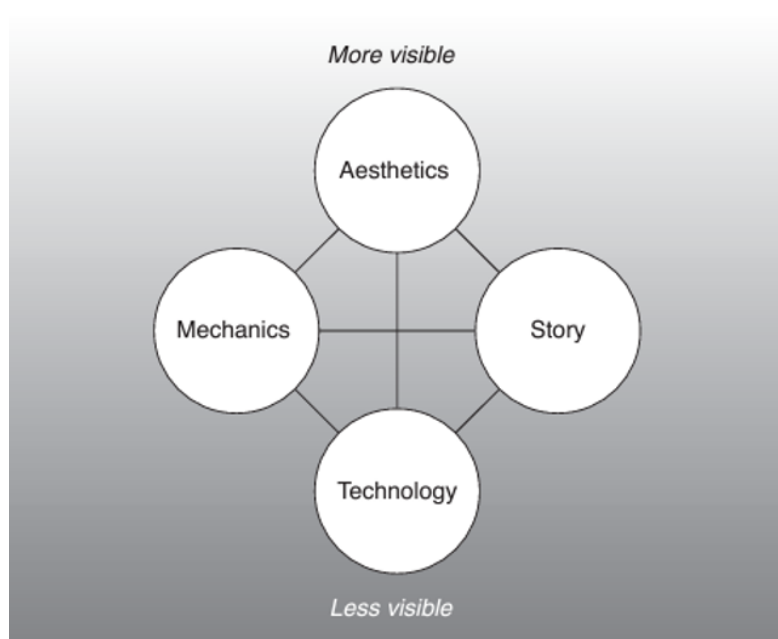
To put this into practice, one way to implement these tactics would be to create levels that become incrementally more difficult, but not so hard that it becomes impossible for the player to pass the level - the player should be kept in the *Flow Channel*. Furthermore, rewards can take on additional significance. For instance, allowing players to exchange points for in-game items such as accessories to customise a game character. Furthermore, a variety of incentives, such as non-materialistic ones such as in-game progress trackers could act as strong motivators, particularly in educational contexts where the player's intrinsic motivation plays a key role (Durin et al., 2019).

#### **2.5.4 Utilising a Game Character and The Tetrad Framework**

In addition to reward systems, the introduction of game characters can significantly increase user motivation, particularly in educational applications, targeting younger users. As highlighted by Koster in *A Theory of Fun for Game Design*: "People tend to dress up game systems with some fiction. Designers put artwork on them that is suggestive of some real-world context." (Koster, 2014, Ch. 5, p.80). This commentary emphasises how fiction in games can be used to frame the game's mechanics in a way that is more relatable and engaging, which can be especially helpful for educational applications, where characters might act out these scenarios to teach lessons or as in the author's example - teach adequate

brushing techniques. Additionally, in *The Art of Game Design*, Jesse Schell discusses the Tetrad (Jesse Schell, 2008, Ch. 4, p. 41) which is a framework that consists of 4 elements of game design - Mechanics, Story, Aesthetics, and Technology. These elements are interdependent and each of them plays a crucial role in the overall design of a game. This Tetrad connection is captured by Figure 2.5. Each element is described in the book as follows:

- **Mechanics:** The procedures and the rules of the game which help describe the goal of the game - how or how not to reach it and how would players be rewarded if they do.
- **Story:** A sequence of events that will unfold in the game.
- **Aesthetics:** How the game "looks, sounds, smells, tastes, and feels"
- **Technology:** Any materials - software, hardware for instance, that make a game possible.



**Figure 2.5:** This figure depicts a framework suggested by Jesse Schell in *A Theory of Fun for Game Design* (Jesse Schell, 2008, Ch. 4, p. 41) - The Tetrad. This framework consists of 4 elements of game design - Mechanics, Story, Aesthetics, and Technology. These elements are interdependent and each of them plays a crucial role in the overall design of a game.

Connecting Raph Koster's insights from "A Theory of Fun for Game Design" with Jesse Schell's Tetrad framework, particularly focusing on the Story and Aesthetics aspects, it can be easily seen how fictional elements (part of Aesthetics) are used to enhance the storytelling component of games. As mentioned above, Koster suggests that fiction can help contextualise the game mechanics, making them more relatable and engaging for users. That indirectly aligns with Schell's idea of how leveraging a compelling *Story* and appealing *Aesthetics* can significantly enhance a player's experience. Moreover, in Chapter 6 of *The Art of Game Design*, Schell explains the elements of the Tetrad work together. For instance, the mechanics of a game should align with and support the story which could be captured by the aesthetics and all made possible through the use of technology. In educational games, using characters that children find appealing can make the learning mechanics more intuitive and effective. The next subsection describes 2 renowned educational applications that utilise the gamification strategies mentioned above.

### 2.5.5 Effective Educational Applications

Besides the applications already mentioned in Section 2.4, there are a couple of prominent applications that demonstrate how utilising game characters and meaningful reward systems can be leveraged in educational contexts. In the context of game characters, Duolingo<sup>17</sup> is one of the most well-known applications for learning languages. As of the end of 2022, Duolingo had over 500 registered learners with 37 million of them being active at least once a month<sup>18</sup>. Duolingo's primary character, *Duo*, serves as a friendly mascot that users can relate to. The mascot is a green, friendly-looking owl that serves as a symbol for the application as well as a personal coach and motivator throughout lessons. Duo appears in reminders and achievements, making learning a new language more feel more personalised and engaging. Moreover, there are several other characters in Duolingo that keep reappearing throughout different lessons, further enhancing the user's experience. Additionally, this is further confirmed by Greg Hartman, the Head of Art at Duolingo, who mentions in a blog post<sup>19</sup> how Duolingo's objective of making their learners spend more time within their application was by introducing this cast of characters, complemented by the green owl mascot - Duo. Similarly, Khan Academy<sup>20</sup>, an organisation that produces short video lessons for students and has over 137 million users signed up<sup>21</sup>, leverages a mastery-based system that allows users to progress at their own pace, ensuring comprehension before advancing. This way, users can choose their personalised path depending to their own preferences.

### 2.5.6 Challenges and Ethical Considerations

Although gamification strategies can offer a plethora of benefits, they also pose several challenges and ethical considerations. The overuse of extrinsic motivation is one of the main concerns. As Raph Koster highlights in Chapter 5 of *A Theory of Fun for Game Design*, the challenge is to find the balance between fun and educational value without sacrificing any of them. To accomplish this, developers should design applications that are engaging but not overly addictive. Additionally, there is a significant ethical concern regarding data privacy and the ethical use of behavioural data collected from users (Kreuter et al., 2018; Harari et al., 2016), especially when those users are children. Maintaining the integrity and trustworthiness of educational technologies depends on making sure that such data is used responsibly.

### Gamification's Effects on User Behaviour

Gamification can significantly influence user behaviour. This is accomplished by encouraging repeated engagement and promoting habit formation. As noted in Subsection 2.5.3 by the author, rewards can seriously enhance user's motivation and engagement by providing them with immediate feedback on their performance. However, gamification can also result in negative consequences such as reduced intrinsic motivation - when users engage in activities to get rewards rather than because learning has intrinsic worth (Antonaci et al., 2019; AlMarshedi et al., 2017). In order to avoid such risks, it is crucial to develop gamified applications that promote a healthy balance between between intrinsic and extrinsic motivations, ensuring healthy and sustainable habit formation and learning outcomes.

<sup>17</sup><https://www.duolingo.com/>

<sup>18</sup><https://www.usesignhouse.com/blog/duolingo-stats#:~:text=Dueling%20has%20over%20500%20million,are%20active%20once%20a%20month.>

<sup>19</sup><https://blog.duolingo.com/building-character/>

<sup>20</sup><https://www.khanacademy.org/>

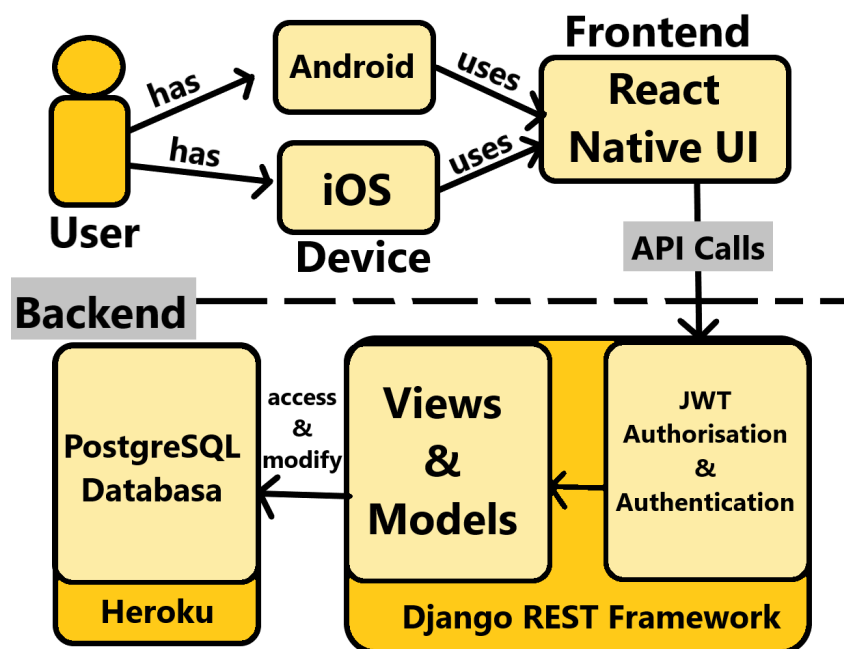
<sup>21</sup><https://www.edweek.org/technology/khan-academy-founder-on-how-to-boost-math-performance-and-make-free-college-education-a-reality/2022/04#:~:text=Among%20the%20most%20popular%20is,million%20users%20across%20190%20countries.>

## Chapter 3

# Technical Review and System Architecture

### 3.1 Introduction to the System Architecture

In this chapter, the author will detail the choices they made in order to create a scalable, maintained application. A simplified version of the relationship between a user with the frontend, and between the frontend with the backend can be seen in Figure 3.1. The frontend, developed with React Native to make the application available across iOS and Android devices, can be accessed by a client using their mobile device. The backend, which leverages the Django REST Framework (DRF), can be reached via Application Programming Interface (API) calls and it communicates with the database server which stores all data stored by the application. The frontend, backend and backend server make up the primary components of the system architecture. This chapter will look at how these components work together and explain how software engineering methods are applied to increase development efficiency.



**Figure 3.1:** A simplified version of the relationship between a user with the frontend, and between the frontend with the backend. The frontend, developed with React Native to make the application available across iOS and Android devices, can be accessed by a client using their mobile device. The backend, which leverages the Django REST Framework (DRF), can be reached via Application Programming Interface (API) calls and it communicates with the database server which stores all data stored by the application.

## 3.2 Backend

### 3.2.1 Framework

Regarding the choice of a framework, suitable for the purposes of the application, there were numerous options. Initially, the author selected Firebase<sup>1</sup>, which is a cloud-based backend computing service, owned by Google, and provides developers with rapid startup without much previous experience. However, shortly after, Firebase had to be replaced by Django<sup>2</sup> - a web framework built on top of Python. There were several reasons for this decision. Firstly, Firebase is free of charge only until a certain point whereas Django is an open-source, free, framework. Additionally, by allowing developers to create and manage more sophisticated database structures, Django offers more flexibility (Holovaty, 2024), which was what was needed for the author's application. Furthermore, Django provides built-in functionalities for numerous challenges when it comes to web development such as:

- **Object-Relational Mapping (ORM):** Instead of writing SQL queries to interact with the database, Django allows developers to complete this task using Python code. Although SQL queries were unavoidable for the author for some actions, like checking an individual user and their related columns in the database, ORM significantly reduced the development time, and additionally enhanced the backend's maintainability and readability, as Python is written in such a way that is very similar to English.
- **Database Schema Migrations:** Django automatically manages database schema changes via migrations. As a result, the student was able to make changes in their models without any data loss. For instance, when adding a new column to a user model, none of the previous users created had to be deleted, as Django provides the option to implement such columns with default values.
- **Security Features:** Security is a critical concern when it comes to developing applications that manage user data. Django provides authentication mechanisms that handle user accounts, permissions, and other authentication-related issues. It also provides built-in protections against several common security threats such as SQL injections<sup>3</sup>, Cross-Site Scripting<sup>4</sup> (XSS), Clickjacking, and Cross-Site Request Forgery<sup>5</sup> (CSRF).

### 3.2.2 Django and RESTful APIs

Besides Django, the student decided to extend Django their backend with the Django REST Framework (DRF) which helps developers build web Application Programming Interfaces (APIs), which are crucial for handling Hypertext Transfer Protocol<sup>6</sup> (HTTP) requests between the frontend and backend. DRF was chosen for its key features which include:

- **Serializers:** The Django REST Framework relies heavily on serializers to make it easier to convert complicated data types into Python data types and vice versa. In essence, the serializer converts the

---

<sup>1</sup><https://firebase.google.com/>

<sup>2</sup><https://www.djangoproject.com/>

<sup>3</sup>[https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)

<sup>4</sup><https://owasp.org/www-community/attacks/xss/>

<sup>5</sup><https://owasp.org/www-community/attacks/csrf>

<sup>6</sup><https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>

Django models into a JavaScript Object Notation<sup>7</sup> (JSON) content type and is suitable for React Native frontend codebase, which is the case for the author's project. Moreover, serializers also process incoming data from requests into formats that are easy to manipulate and validate (make sure that the incoming data adheres to the predefined data structures and rules) within the Django REST framework. The example in Figure 3.2 shows a practical example of JSON formatted login data type, containing pairs of keys and values, and the corresponding Python dictionary which will be then used by the backend to perform operations.

JSON :	Python :
{	data = {
"email": "example@email.com",	"email": "example@email.com",
"password": "eXampl!Passw00rD"	"password": "eXampl!Passw00rD"
}	}

**Figure 3.2:** Example of data conversion from JSON to Python using Django REST Framework serializers. The JSON object on the left represents data received from the server, which is then transformed into the Python dictionary on the right, ready for backend operations

- **Views:** When HTTP requests are sent to views (also known as endpoints) in DRF, they are used for CRUD<sup>8</sup> (create, read, update, and delete) operations. For DRF, they correspond to *POST*, *GET*, *PUT*, and *DELETE*. Applications that use these methods are called *RESTful APIs*. They control the logic that manages how a user's request should be handled, process it, and return a response back to the user. DRF offers two approaches to defining views. *Function-Based Views* are the first type; they are appropriate for basic applications and are created as Python functions. The second one is *Class-Based Views*, which are defined as Python classes, and offer a more complex structure, and are what the author utilised for their application. Figure 3.3 shows the pseudocode for a Class-Based view which allows an authenticated user to delete their account. If successful, the view will respond with an *HTTP 204 No Content* status code back to the user, and if not, either an *HTTP 404 Not Found* (user does not exist) or an *HTTP 400Bad Request* (any unexpected exceptions).
- **Browsable APIs:** Browsable APIs are a feature that is unique to DRF. Every API endpoint has a webpage that is generated, as seen in Figures 3.4 and 3.5. This way, the developer can directly interact with the API, making it possible to test and debug backend code before it is used by the frontend. In the example provided in the figures, the author is testing the *'/application/signin/* endpoint which accepts JSON formatted data-email and password keys with their corresponding values, and executes a *POST* request to the server. Based on different factors such as internet connection, valid input, and user existence, the server will return an appropriate HTTP response. For instance, if the action shown in 3.5 is carried out, the server is expected to return an *HTTP 401 Unauthorized* response if the credentials provided do not match any existing users.

<sup>7</sup><https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON>

<sup>8</sup><https://www.codecademy.com/article/what-is-crud>

```

1: Class: DeleteUser
2: Require: JWT Authentication, Authenticated User
3: function DELETE(request)
4:     user ← request.user                                ▷ Get the authenticated user
5:     if user exists then
6:         user.delete()                                ▷ Attempt to delete the user
7:         return Response(status 204)                    ▷ Success
8:     else if user does not exist then
9:         return Response("User not found", status 404)
10:    else
11:        return Response("Unexpected error occurred.", status 400)
12:    end if
13: end function

```

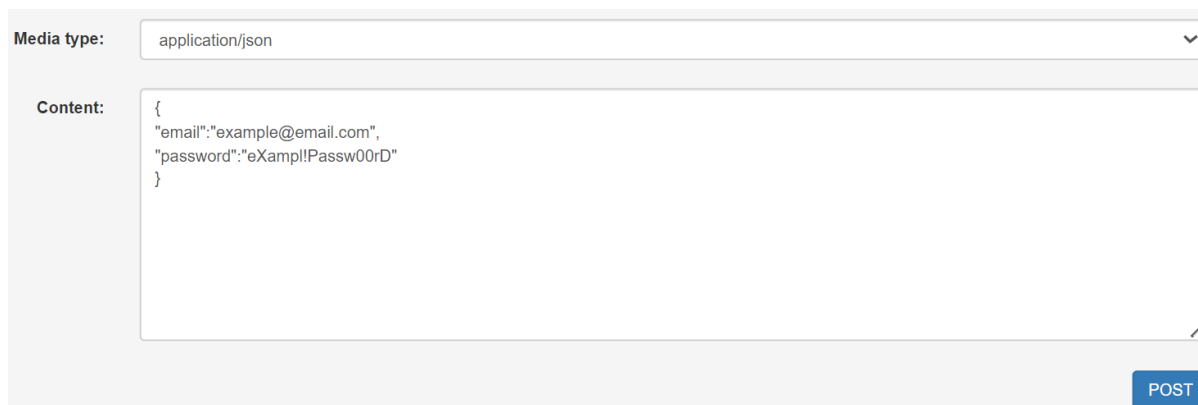
**Figure 3.3:** Pseudocode showing a Django REST Class-Based View for deleting a user. It allows an authenticated user to delete their account. If successful, the view will respond with an *HTTP 204 No Content* status code back to the user, and if not, either an *HTTP 404 Not Found* (user does not exist) or an *HTTP 400Bad Request* (any unexpected exceptions)



**Figure 3.4:** Testing the `/application/signin/` endpoint which accepts JSON formatted data-email and password keys with their corresponding values, and executes a *POST* request to the server

- **Authentication and Permissions:** Authentication and Permissions: For any application, authentication is a pivotal part that not only verifies one's identity, but it additionally ensures that only authorised users (ones given permission) can access particular endpoints from the backend or perform specific operations. The Django REST framework includes several methods to authenticate users such as either session or token authentication. The author chose to employ the JSON web

(JWT) tokens in order to create an application that has a secure way of transporting sensitive information. The JWT tokens work by utilising a combination of access and refresh tokens. In comparison to the refresh token, the access token that is designed to be created from it has a much shorter validity. Usually, when the access token expires, users are directed to the login page in order to reauthenticate. Although tokens, like every other security measure, could be compromised and depend on a secret key that also should not be exposed, they provide more robust and secure handling of data due to their limited lifetime and transmission security (they can only be transmitted over secure Hypertext Transfer Protocol Secure (HTTPS) connections and are secured by a digital signature). Additionally, by design, DRF applications provide developers with Permission classes that determine what actions an authenticated user is allowed to execute. One of those classes is the `'isAuthenticated'` permission which only grants access to authenticated users. Consequently, by implementing both tokens and permissions, the author's application is kept secure and protected against unauthenticated or unauthorised user manipulation over the backend.

The image shows a web interface for a browsable API. At the top, there is a dropdown menu labeled 'Media type:' with 'application/json' selected. Below this is a large text area labeled 'Content:' containing a JSON object: 

```
{  "email": "example@email.com",  "password": "eXampl!Passw00rD"}
```

. At the bottom right of the interface is a blue button labeled 'POST'.

**Figure 3.5:** Browsable API interface illustrating how to correctly submit a *POST* request with user credentials in JSON format

### 3.2.3 Database Management

The author decided to use PostgreSQL<sup>9</sup> and SQLite<sup>10</sup> as the databases for managing the data of the application. The former, also commonly referred to as Postgres, is a free and open-source relational database management system (RDBMS) that is widely used for production databases due to its reliability and robustness. Furthermore, PostgreSQL includes features like Optimistic Concurrency Control<sup>11</sup> (OCC), which enables concurrent read requests from users without the need for read locks (as opposed to only one user being to execute read operation at a time). Although there were alternatives, PostgreSQL gives more functionalities than an SQL Server, including support for the JSON data type, which is the author uses for their application. Nevertheless, the primary factor that influenced this decision was also the choice of service to deploy the backend, Heroku<sup>12</sup>, which the author will speak about in the next subsection. This is due to the fact that Heroku<sup>12</sup> utilises a managed PostgreSQL database. For local development testing, the author chose to use SQLite, which is a lightweight database engine written in the C programming language. It was not utilised for production as it is limited in its scalability and

<sup>9</sup><https://www.postgresql.org/>

<sup>10</sup><https://www.sqlite.org/>

<sup>11</sup>[https://en.wikipedia.org/wiki/Optimistic\\_concurrency\\_control](https://en.wikipedia.org/wiki/Optimistic_concurrency_control)

<sup>12</sup><https://www.heroku.com/>



concurrency. However, SQLite allows for a quick startup process which was ideal for local testing on the student's machine before deployment on the production database.

### 3.2.4 Backend Deployment

The author decided to use Heroku<sup>13</sup>, a widely used platform as a service<sup>14</sup> (PaaS), to deploy the backend application. Heroku enables its users to "build, run, and operate applications entirely in the cloud" and supports a number of programming languages, including Python and the Django REST Framework. Heroku is an affordable choice even though it does not offer a free tier version. Additionally, apart from utilising a managed Postgres database, this service is suitable for projects that will eventually need to scale up. After evaluating costs, Heroku proved out to be one of the most suitable choices. The *GitHub Student Developer Pack*<sup>15</sup> was utilised by the author as it provides students with a year of free access to the Heroku platform. Furthermore, Heroku provides an abstraction of the infrastructure layer, allowing developers to concentrate on the code rather than service management, in contrast to other platforms like Amazon Web platforms<sup>16</sup> (AWS) or Google Cloud Platform<sup>17</sup> (GCP).

### 3.2.5 Environment Variables

Environment variables<sup>18</sup> give developers the security they need to safeguard the configuration settings of their backend. They keep important information, such as secret keys (for JWT encryption and cryptographic signing) and database URLs required to connect to the database, in an `.env` file that resembles Figure 3.6 and is never pushed to a version control system.

```
SECRET_KEY = secret_key_value
DATABASE_URL = url_to_the_database
JWT_SECRET = for_jwt_tokens_encryption
```

**Figure 3.6:** Example of what an Environment *.env* file typically looks like

### 3.3 Frontend

The choice of a programming language to develop the frontend and its functionalities was crucial. It was decided that a cross-platform language should be employed to provide a consistent UI across different devices and to avoid the necessity of learning platform-specific languages for iOS (Swift, Objective-C) and Android (Kotlin, Java, C++) as the author was limited in their development time. There were several solutions when it came to selecting one suitable for the purposes of the author's application, such as React Native, Flutter, Xamarin, and Ionic. The author made the choice to utilize React Native Command Line Interface (CLI), which is a JavaScript-based framework. The author also examined choices such as Flutter; however, the choice was made based on React Native's wide community support

<sup>13</sup><https://www.heroku.com/>

<sup>14</sup>[https://en.wikipedia.org/wiki/Platform\\_as\\_a\\_service](https://en.wikipedia.org/wiki/Platform_as_a_service)

<sup>15</sup><https://education.github.com/pack>

<sup>16</sup><https://aws.amazon.com/>

<sup>17</sup><https://cloud.google.com/>

<sup>18</sup><https://circleci.com/docs/env-vars/>

and JavaScript foundation, which was already somewhat familiar to the author. Choices like Flutter would have required more development time and a steeper learning curve. Nevertheless, React Native is still considered to be one of the best cross-platform frameworks for developing mobile applications due to its great performance (Gülcüoğlu et al., 2021). Additionally, React Native CLI offers seamless integration capabilities for establishing communication with a Django backend, used by the author, through libraries such as Axios, to make sending and retrieving data straightforward. The subsections below will discuss the reasons why the student picked this framework, exploring the advantages of CLI over Expo CLI, which are both options that developers can choose between when developing mobile applications.

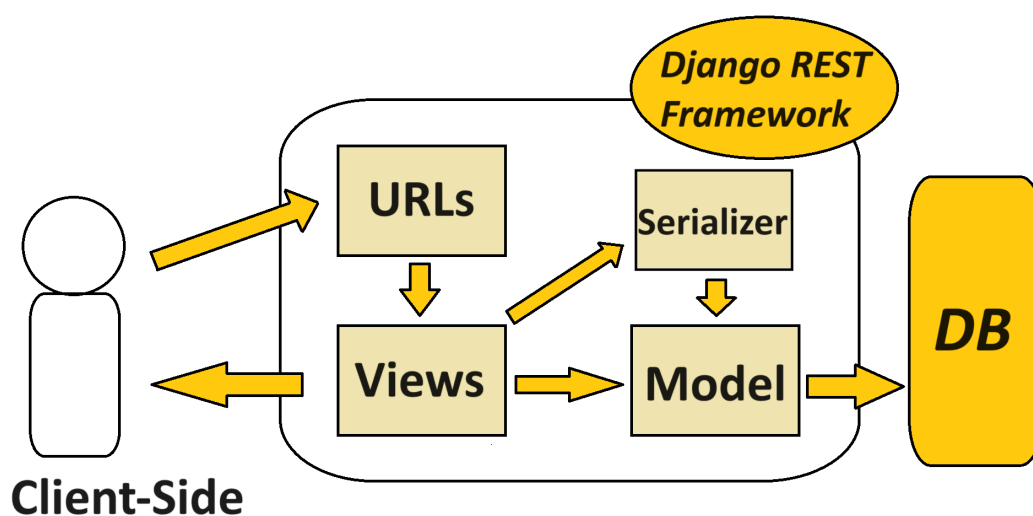
### **3.3.1 Initial Approach**

The first approach towards developing the codebase that the author undertook was utilizing React Native Expo CLI. Expo is almost identical to React Native CLI but provides an abstraction layer that simplifies setting up a project and aids developers in creating prototypes faster. Additionally, some of the main advantages of Expo include over-the-air updates, as Expo provides its users with the Expo Go application, which is a free, open-source sandbox for testing React Native code. Using this approach, the author was able to see any changes made in the code immediately on their device. Another advantage of Expo is the Expo Application Services (EAS), which allow developers to simplify the building and deployment process of an application before submitting it to the App Store or Google Play, whereas in React Native CLI, this has to be done manually. Furthermore, Expo provides access to numerous pre-built libraries and APIs, making it easier to add advanced features such as push notifications and maps.

### **3.3.2 Final Approach**

However, Expo was not the author's final choice of a framework. Over time, the author came across some of its disadvantages. A disadvantage of Expo is the amount of time it takes for the EAS free-tier services to build an .apk Android file (starting from at least 30 minutes). Another of the major influences was the restricted access to native modules, which pose limitations when certain third-party libraries have to be included. For instance, to handle sensitive user data, one of the best options the author could use was React Native Encrypted Storage, which is not offered by Expo. Instead, Expo provides the SecureStore library, which has limited functionality in comparison, missing some functions such as wipe(), which deletes the contents of the encrypted storage in a single line, as opposed to having to implement a function to do this. Additionally, with Expo, developers are not able to access low-level device features like advanced Bluetooth settings, camera, or Augmented Reality. To mitigate the future risk of having to switch from React Native Expo CLI to React Native CLI, which would require a substantial amount of time and effort, the author decided to switch frameworks. All the factors mentioned previously contributed to this decision as React Native CLI offers greater flexibility and control over the application's frontend. Additionally, the author tested the difference between a simple, single-screen application displaying "Hello World" for both frameworks, as Expo is known to produce applications with larger bundle sizes. This was confirmed by the author as well; when they built the application over both frameworks, Expo produced a file almost double the size of that from React Native CLI. Although there were no explicit studies directly comparing the advantages of one framework versus the other, given the nature of the student's project and the issues they encountered personally, as well as reading what the community says on websites like Stack Overflow, the decision to use solely React Native CLI was taken.

### 3.4 Backend Architecture



**Figure 3.7:** The diagram presents a client-server (frontend-backend) interaction within the Django REST Framework. Here the client requests are received by the URL dispatcher, which selects the appropriate view from the backend side. The view interacts with the model utilising the CRUD operations. Upon completion, a JSON format serialized response is returned back to the user.

The backend model utilises a monolithic application approach - all backend functionality resides within a single application. This approach simplified the deployment and the management of the author's codebase. The choice of this structure, as opposed to the microservice one for instance, was due to the reason that the project currently does not require a complex structure and a monolith offers an efficient structure without unnecessary overhead. The author also structured their code in a modular and maintainable way, such that, if needed in the future, they can easily transition to a more compound structure. The Django REST Framework traditionally follows a Model-View-Controller (MVC) design pattern. The author adapted a version of the MVC pattern, specifically a Model-View-Serializer pattern, which is tailored for building RESTful APIs, which are crucial for the communication between the backend and React-Native CLI frontend. An extensive description of this architecture's components will be now provided:

Field	Type	Description
first_name	CharField	First name of the user
last_name	CharField	Last name of the user
email	EmailField	Unique identifier for user login, used for authentication
thumbnail	ImageField	Optional profile picture of the user
total_brush_time	IntegerField	Total time the user has spent brushing their teeth
current_level	IntegerField	The user's current level in the application
current_level_xp	IntegerField	The user's current experience points within their level

current_level_max_xp	IntegerField	The maximum experience points for the current level
image_id	IntegerField	An identifier for the character's image
current_streak	IntegerField	Current consecutive days of app engagement
max_streak	IntegerField	The maximum streak of consecutive days of app engagement
total_brushes	IntegerField	Total number of times the user has brushed
parent_pin	CharField	Pin to access the parent interface
is_pin_set	BooleanField	Indicates whether a parent pin is set
last_active_date	DateField	The last date the user was active
last_active_morning	DateTimeField	The last morning the user was active
last_active_evening	DateTimeField	The last evening the user was active
streak_morning	IntegerField	Current morning streak of the user
max_streak_morning	IntegerField	Maximum morning streak of the user
streak_evening	IntegerField	Current evening streak of the user
max_streak_evening	IntegerField	Maximum evening streak of the user
total_brushes_morning	IntegerField	Total morning brushes by the user
total_brushes_evening	IntegerField	Total evening brushes by the user
total_brushes_days	IntegerField	Total days the user has brushed
percentage_morning	FloatField	Morning brushing percentage
percentage_evening	FloatField	Evening brushing percentage
character_name	CharField	Name of the character associated with the user
is_char_name_set	BooleanField	Indicates whether a character name is set for the user

**Table 3.1:** Description of all the fields associated with the User Model implemented by the author and their respective description.

- **Models:** They remain the core of the architecture and reflect the application's data structures and logic. Models deal directly with the database, which is in charge of transaction management and data integrity as well as the CRUD operations. Table 3.1 shows the User Model implemented by the author with all of the data fields associated with it. A more detailed description of the model's implementation can be found in Chapter 4.
- **Views:** To build on the description in Subsection 3.2.2, Views typically are similar to controllers in an MVC architecture. However, they often contain logic for handling user input (for instance password validation) and data processing, masking the usual MVC separation of such.

Endpoint	URL Path	Description
Sign Up	/signup/	Registers a new user
Sign In	/signin/	Logs in existing users
Is Signed In	/authenticated/	Checks if the user is authenticated
Sign Out	/signout/	Logs out an authenticated user

Delete User	/delete/	Deletes an authenticated user's account
Token Refresh	/token/refresh/	Refreshes the JWT access token
Update Time	/update_brushtime/	For total brush time
Set Parent Pin	/setPin/	Sets the parent control PIN
Is Pin Set	/isPinSet/	Checks if the parent control PIN is set
Check Parent Pin	/checkPin/	Validates the parent control PIN
Update Level	/levelUp/	Updates the user's level
Update User Level XP	/updateUserXP/	Updates the user's experience points
Update Max Level XP	/updateCurrentLevelMaxXP/	Updates the maximum XP for the current level
Mini Shop Photo	/miniShop/	Handles image selection in the mini-shop
Update Streak	/updateStreak/	Updates the user's brushing streak
Update Activity	/updateActivity/	Logs a new brushing activity
User Activities	/activities/	Retrieves all user activities
Set Character Name	/updateCharacterName/	Sets the name of the user's character

**Table 3.2:** Connection between Views and URLs in the author's application. The first column entitles the specific endpoint (view) whereas the second one provides the specific URL path leading to it. The third column describes each view respectively.

- **Serializers:** As described in Subsection 3.2.2, the serializers' role is to provide a way of connecting models and views, by utilising the JSON format, which allows for more complex structures to be easily converted and manipulated.
- **URLs:** They are the routing mechanism to direct incoming HTTP requests to the appropriate view based on the URL pattern. They act as controllers in a usual MVC architecture, deciding which view should handle which request. Table 3.2 describes all of the views in the author's application as well as their respective URL paths.

This MVS architecture can be described as follows and is depicted in Diagram 3.7:

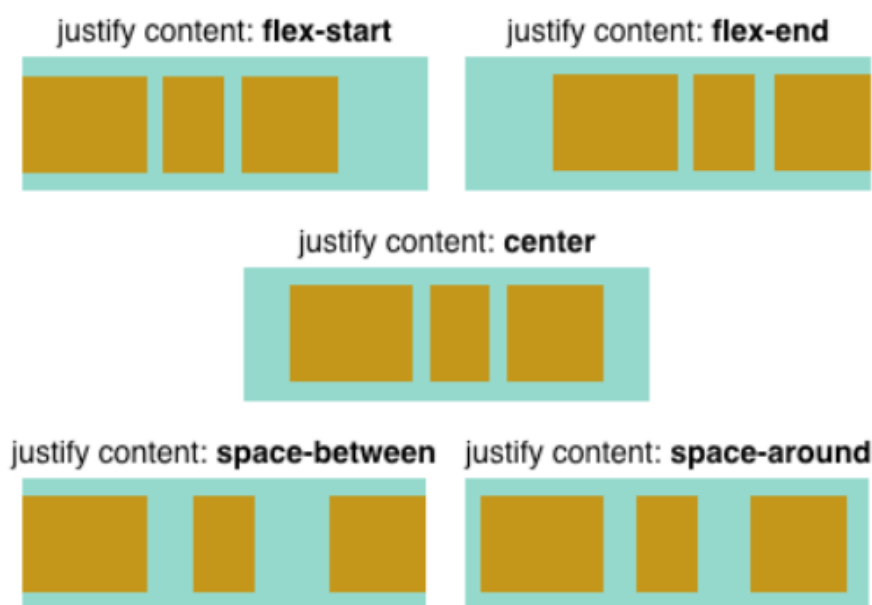
- When the client/frontend makes a request, this request is sent to the appropriate view in the back-end by the URLs.
- The particular view interacts with the application's model to retrieve, update, or delete specific data.
- The serializer then converts this data to and from the JSON format which is required by the client/frontend.
- Lastly, the serialized response is sent back to the client/frontend.

## 3.5 Frontend Architecture

As mentioned in section 3.3, React Native CLI was chosen by the author for its extensive community support and cross-platform maintainability. Furthermore, this framework utilises the declarative programming model which, as opposed to the imperative model, does not require the developer to specify

the exact flow of control for their application and simplifies the data flow of the application. This approach allows for enhanced maintainability and scaling up of the application over time. The React Native frontend architecture can be split into several components, each of which has the responsibility to handle different layers of the application's functionality:

- **JavaScript Thread:** The main component for executing the application's business logic (specific rules and calculations that make applications function correctly), which mostly utilises the JavaScript and React framework. The JavaScript Thread manages the user interactions and coordinates the application's flow.
- **Native Modules:** They use the platform-specific language (e.g. Java for Android and Swift for iOS) and act as intermediaries to a device's core features such as camera, and Global Positioning System (GPS), to provide users with native-quality experience.
- **UI Thread:** The application's visual components are rendered by the UI Thread which is unique to each platform (Android and iOS). It provides users with seamless performance by taking instructions from the JavaScript Thread and turning them into the views that the users interact with.
- **Shadow Thread:** The Shadow Thread<sup>19</sup> uses the Yoga layout library to decide where to place items in the application's background. This approach makes sure that the application's layout is handled in an efficient manner without interrupting the UI Thread while allowing developers to create responsive designs, responsive to changes in the design. An example use of the Yoga library can be seen in Figure 3.8, which shows 5 different methods to justify content on a screen.



**Figure 3.8:** Example ways to position elements on a screen using React Native's Yoga library<sup>20</sup>.

These components and their roles within React Native and its architecture can also be seen in Table 3.3, which provides shorter, easy-to-understand descriptions. The JavaScript Thread acts as the core of

<sup>19</sup><https://www.kodeco.com/530-yoga-tutorial-using-a-cross-platform-layout-engine>

<sup>20</sup><https://www.kodeco.com/530-yoga-tutorial-using-a-cross-platform-layout-engine>

the application to handle the sequence of user interactions. It communicates with the UI and Shadow Thread via a *bridge* mechanism which is a bidirectional channel that allows commands and requests from the JavaScript Thread to be sent, and then either translated into native views by the UI Thread or layout calculations by the Shadow Thread. This interaction between components ensures the seamless user experience as JavaScript performs all of the heavy lifting of logic and state management while the UI and Shadow Thread remain focused on their respective tasks. This bridging concept allows for non-blocking UI, which means that long-running JavaScript tasks cannot cause the UI to stutter or freeze since they are processed separately from the UI Thread. In the next section, the author will focus on the relationship between the React Native CLI frontend and Django REST backend.

Component	Simple Description	Responsibilities	Key Technologies
JavaScript Thread	Where the app's main code lives	Running the app logic	JavaScript, React
Native Modules	Connectors to the devices's features	Making use of the phone's camera, GPS, etc.	Java for Android, Swift for iOS
UI Thread	The main thread responsible for handling user interface updates and rendering	Showing the app interface to the user	Native platform code
Shadow Thread	Layout calculation	Compute UI layout for different screens (where things go on the screen)	Yoga layout library

**Table 3.3:** React Native Frontend Architecture Components

## 3.6 Frontend-Backend Relationship

The relationship between frontend and backend is crucial for seamlessly integrated application that provides exceptional user experience. The next subsections will discuss how the interplay between The React Native and DJANGO REST Frameworks in the above mentioned architecture, ensures a fluid and intuitive interface for end-users.

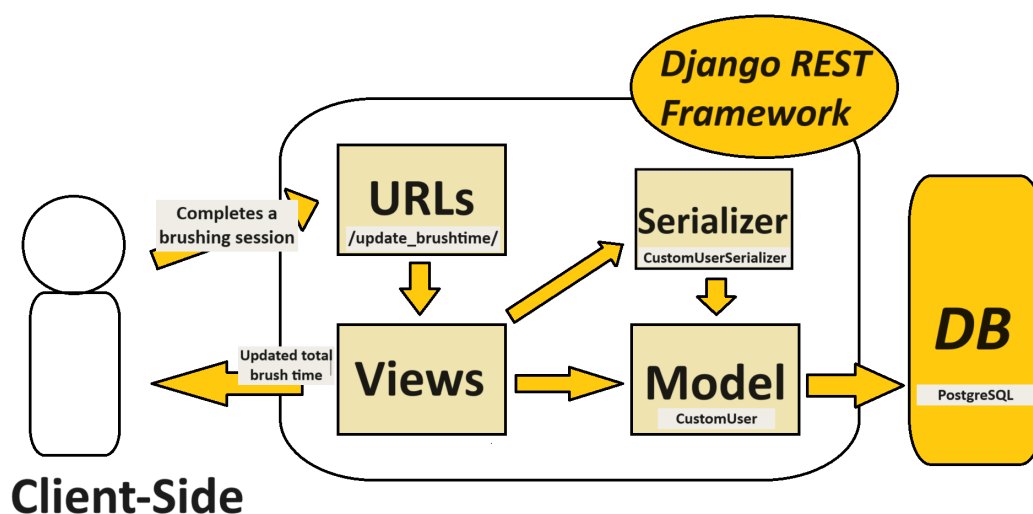
### 3.6.1 Data Exchange and API Communication

Frontend and backend communicate through API calls. These API calls are HTTP requests from the frontend to the defined endpoints mentioned in Table 3.2. Upon receiving a specific request, the Django REST backend performs the necessary operations and returns a structured *JSON* response to the frontend which is then used to render the updated state to the user interface.

### 3.6.2 Practical Example

To demonstrate how this relationship works in practice, a modified version of Figure 3.7 is shown in Figure 3.9. In the figure, the user completes a brushing session, after which the frontend sends an update request to the `/update_brushtime/` endpoint which is then handled by the URL dispatcher to select the appropriate view for this operation. The view interacts with the CustomUser Model, modifying the

appropriate fields and updating the PostgreSQL database. Upon completion, a JSON serialized response, created by the CustomUserSerializer is returned back to the frontend to display the total time a user has brushed their teeth for.



**Figure 3.9:** The diagram presents a client-server (frontend-backend) interaction within the Django REST Framework in the author’s application. Here the user completes a brushing session, after which the frontend sends an update request to the `/update_brushtime/` endpoint which is then handled by the URL dispatcher to select the appropriate view for this operation. The view interacts with the CustomUser Model, modifying the appropriate fields and updating the PostgreSQL database. Upon completion, a JSON serialized response, created by the CustomUserSerializer is returned back to the frontend to display the total time a user has brushed their teeth for.

### 3.7 Implementation

The implementation and design choices taken when creating both the React Native CLI and Django REST backend architectures, can be found in chapter 4 which describes their implementation in detail. In the next section the author will discuss the developmental approach they undertook while creating the application.

### 3.8 Software Engineering Practices

The Agile development methodology was crucial for the project’s success. The term "agile software development" refers to methods for creating software. Agile’s main objective is to produce high-quality software on a regular and consistent basis through the use of short development cycles, or sprints, that typically last one to four weeks. However, the author’s project was an individual project and hence they modified this Agile approach, utilising some of its methodologies like Scrum, Lean, Kanban and Extreme Programming (XP), to cater to a single-person development team in collaboration with a project client and supervisor. The subsections below detail this approach and its principles.

#### 3.8.1 Scrum Adaptation

Scrum is a framework within the Agile methodology that puts an emphasis on team work, accountability and iterative progress towards a clear objective. The process is split into sprints, which are short cycles lasting 2 to 4 weeks, where there are specific tasks that need to be completed and reviewed. The Scrum



framework utilises daily meetings, called stand-ups, sprint reviews and sprint planning in order to facilitate communication and faster, more efficient problem-solving. The author adapted Scrum to their individual project as follows:

**Sprint Planning:**

Given the time frame for the project, the student used their weekly meetings with the client to plan and discuss the sprint goals, each of which lasted a week. At those meetings, tasks that needed to be completed within the following week were carefully planned out.

**Daily Scrum:**

Instead of a daily scrum (stand-up) which is done in a team environment, the author followed a modified approach. At the start of each day they wrote down each task that needs to be completed, based on the current sprint, followed by a brief self-check in the end of the day to keep track of their goals. Tasks that had not been completed were transferred to the next day or next sprint for the scenario where the daily scrum was in the end of a sprint.

**Sprint Reviews:**

At the beginning of each client meeting, the student discussed what they had accomplished during the previous sprint. Additionally, they sought feedback on whether the tasks they completed were in alignment with the client's expectations in order to adjust the sprint for the following week.

**Sprint Retrospective:**

The meetings with the student's supervisor were utilised to reflect on the technical part of the application-what worked well, what did not, and what could have been improved. Those meetings proved to be invaluable to the overall development process.

**3.8.2 Lean Adaptation**

In Agile, the term Lean refers to reducing waste in all its forms, optimising efficiency, and optimising production throughout the software development process. Its foundations are found in the Lean Manufacturing concepts that Toyota and other Japanese automakers pioneered.

**Eliminate Waste:**

The author constantly evaluated their progress and objectives-which ones had to take priority, and which ones, even if considered somewhat important had to be either eliminated completely or left behind for development in the future. The student considered those options initially, and after consulting with the client and supervisor, final choices were made.

**Streamline Processes:**

The author frequently analysed their development process and identified areas they experienced challenges. Those areas were targeted by advising with the project's supervisor, additional reading and streamlining them early on to avoid potential future risks.

**3.8.3 Kanban Adaptation**

The Kanban methodology refers to visualising work, minimising work that is progress and streamlining work. To achieve this task, developers typically use a Kanban Board as seen in Figure 3.10. Kanban Boards typically have 4 columns:

- **Backlog:** 'To-do' tasks that have to take priority into the next sprint.

- **Doing:** Tasks that are currently in progress.
- **Review:** Dedicated to tasks that need to be evaluated, to ensure work of high quality.
- **Done:** Tasks that have been completed and meet all current requirements.



**Figure 3.10:** Kanban Board<sup>21</sup> Visualisation: The board is divided into four columns: 'Backlog' for upcoming tasks, 'Doing' for active work, 'Review' for quality checking, and 'Done' for completed tasks. Kanban Boards are used to visually track workflow progression.

The author utilised this approach by performing the following tasks:

#### **Visual Workflow:**

While the student did not specifically use a Kanban Board, as briefly mentioned in Subsection 3.8.1 about the Scrum Adaptation, a daily Scrum to-do list was utilised. Additionally, in the beginning of the project, the author listed down all of the primary and secondary objectives for the application as well as some further functionalities that could be implemented if time permitted.

#### **Limit Work in Progress:**

Similar to what the Kanban Board suggests - minimise work that is in progress, the author firstly prioritised the primary objectives for the application, followed by the secondary, in order to focus on tasks that are more important.

#### **Continuous Improvement:**

The project's supervisor and client feedback were utilised to continuously optimise the student's workflow and focus on specific tasks during sprints.

### **3.8.4 Extreme Programming (XP) Adaptation**

#### **Test-Driven Development:**

The author thoroughly tested the backend and frontend functionality interactively and exploratory. Every time a view had been created or a new column was added to the database, extensive testing through the Browsable API, mentioned in Subsection 3.2.2, was conducted. The author would test different scenarios

<sup>21</sup>[https://en.wikipedia.org/wiki/Kanban\\_board](https://en.wikipedia.org/wiki/Kanban_board)

and inputs and see how the server reacts. Additionally a mix of integration and functional testing were executed using Django's testing framework and Django REST Framework. Such a test can be seen on Figure 3.11 which shows a pseudocode for testing whether a user can successfully sign up, using correctly formatted credentials. The test expects an *HTTP\_201\_CREATED* response which indicated that the user creation has been successful.

- ```

1: Component: TestSignup
2: Require: Django Test Case, Django REST Framework's APIClient
3: function TESTSIGNUPSUCCESS
4:     Setup: APIClient to create client instance
5:     Data: { first_name: "Test", last_name: "User", email: "test@test.com", password: "password1!D" }
6:     Response: CLIENT.POST("/application/signup/", Data)           ▷ Post data to signup endpoint
7:     Assert: ASSERTEQUAL(Response.status_code, 201)             ▷ Check if HTTP 201 status code is
    returned
8: end function

```

**Figure 3.11:** Using Django’s testing framework and DRF’s APIClient, this function tests whether a new user can register successfully through the application’s signup endpoint. In this example, the test expects an *HTTP\_201\_CREATED* response which indicates that the user has been created with the appropriate data fields.

For the Frontend the author integrated some Jest unit and snapshot tests as Jest is known for its simplicity and is widely used by the React Native community. They ensured that the application's UI does not change unexpectedly. One such test can be seen on Figure 3.12 which shows a pseudocode that tests whether the *Loading* screen is rendered as expected.

- ```

1: Component: AppNavigationComponentTest
2: Require: Jest, React Testing Library, React Navigation
3: function TESTINITIALLOADINGSTATE
4:   Mock: USEGLOBALLY to return { initialised: false }
5:   RENDER(<App>)                                ▷ Render the App component
6:   Assert: QUERYBYTEXT("Loading") is not null    ▷ Check if loading screen is displayed
7: end function

```

**Figure 3.12:** Pseudocode for testing whether the rendering of the "Loading" screen will be successful, using Jest and React Testing Library.

### Frequent Releases:

The student aimed to deliver working prototypes of the project to the client as often as possible, allowing for regular feedback and adjustments aligned with the client's expectations.

## Refactoring and Continuous Integration:

To ensure code maintainability and scalability, the author regularly refactored their code to ensure code readability and quality.

### 3.9 Risk Assessment and Project Timeline

### 3.9.1 Risk Assessment

During the development of any software application, especially one focused on pediatric dental hygiene utilising gamification strategies, multiple risks need to be identified and mitigated in order to ensure a

successful delivery of the final product. This was a particularly critical part in the student's case, since they were the sole developer of the application. They initially developed a risk assessment for their Project Plan, which had since been extended as seen in Figure 3.4. The table contains numerous risks that had to be considered, what the probability of them happening was, how big of an impact they would have made and the strategies implemented to mitigate such risks. The author will discuss whether they encountered any of them in the Chapter 7 - Discussion.

Similarly to the Risk Assessment, the author created a Gantt Chart, that demonstrates an approximate estimate of the project time frame. This Gantt Chart has been extended since and is depicted in Figure 3.13. It shows all of the development cycles followed by the author. As mentioned in Section 3.8, the author followed weekly sprints. Those sprints consisted of:

- A 2 hour meeting with the client to discuss the current progress on the application.
- Planning out the objectives for the following sprint.
- Evaluating the biggest potential risks for that sprint and addressing them first.
- Development of the application's code.
- Testing out each component created.

*Iterative Approach* is another name for the method followed by the author which consists of incremental and iterative improvement through sprint cycles. This method breaks the development process into smaller, manageable iterations, allowing for continuous testing, evaluation, and adaptation at each stage. It's more flexible and better suited for projects with evolving requirements which was the case for the author's client-based application. The other prominent approach is the *Waterfall Approach* which is a linear and sequential method where each phase of development must be completed before moving to the next, with no overlap between phases. It's suitable for projects with fixed requirements and where changes are unlikely which was not the case for the student as their project required for regular changes based on the client's feedback and project's time constrained frame.

<b>Risk Description</b>	<b>Impact (Low/Medium/High)</b>	<b>Probability (Low/Medium/High)</b>	<b>Mitigation Strategies</b>
Technical Incompatibilities	High	Medium	Test all of the technologies in the beginning of the development. Research alternative technologies. Code modularity - be able to switch components easily. Early prototyping .
Misinterpretation of Client Requirements	High	Medium	Weekly client meetings. Clear and regular communication. Be willing to compromise where possible.
Data Security and Privacy Concerns	High	High	Implement best security practices according to DRF official documentation.
Underestimation of Timelines	High	High	Prioritise primary objectives first. Split the project into achievable sprints. Allow time for mistakes.
Dependency on Third-Party Services	Medium	Medium	Ensure they are all coming trusted and well tested and maintained sources.
Technical Skill Gaps	Medium	High	Spend time in the beginning of the project to learn unfamiliar technologies and frameworks.
Hardware/Software Failure	High	Low	Regular backups and use of version control systems.
Legal and Ethical Issues for User Evaluation	High	Medium	Ensure all surveys are ethically approved before distribution and provide clear participant information forms.

**Table 3.4:** Extensive Risk Assessment for Application Development showcasing the biggest potential risks that are related to the author's application, as well as their potential impact and probability to be encountered and mitigated.

3.9.2 Project Timeline and Development Cycles

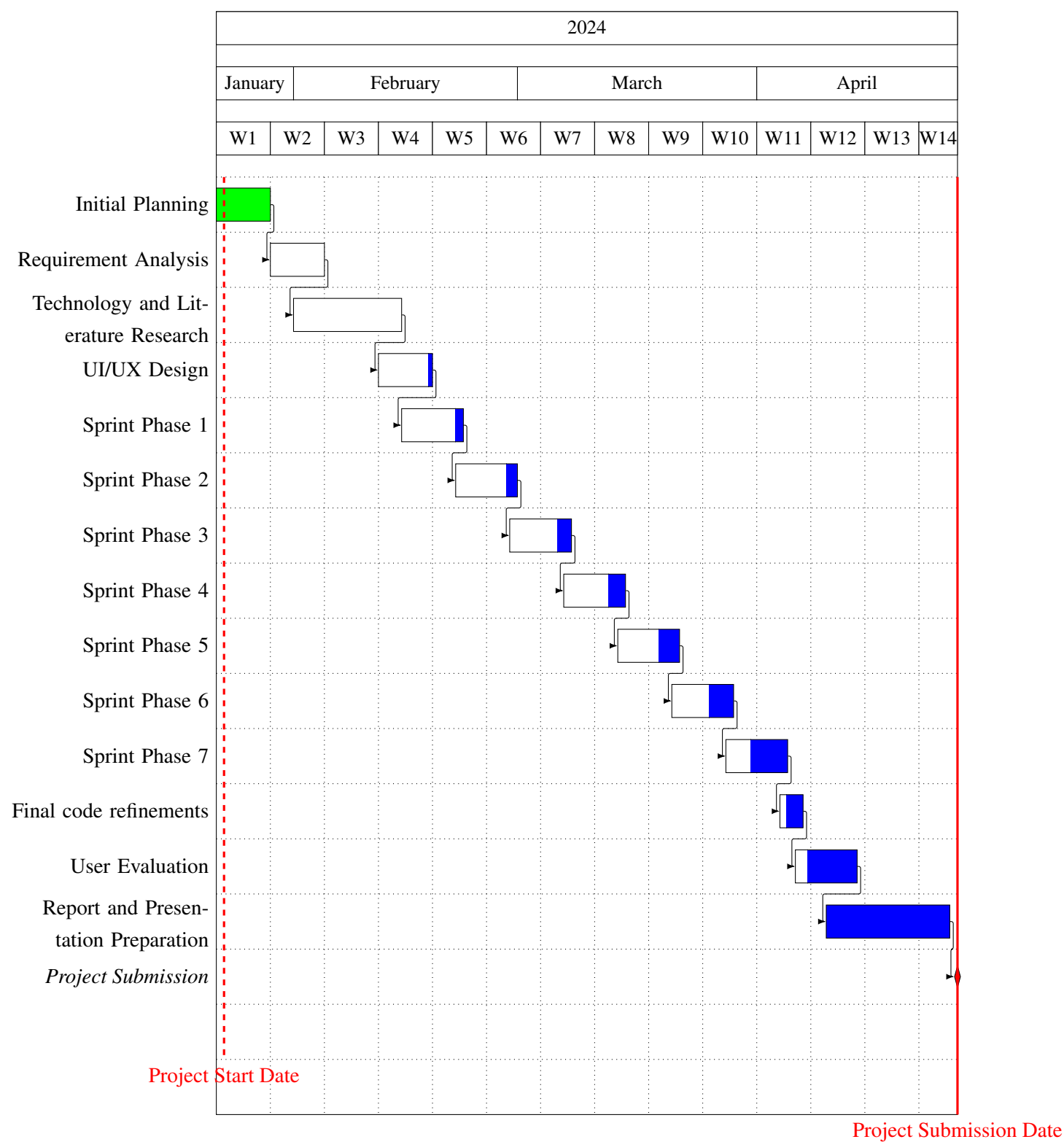


Figure 3.13: Project Gantt Chart depicting the proposed timeline for the author’s developent cycles.

## Chapter 4

# Implementation

### 4.1 Development Environment Setup

In this section, the author will describe selection of the development environment to optimise productivity and effectively manage the application's lifecycle.

#### 4.1.1 Integrated Development Environment (IDE) and Code Editors

For the purposes of this project, the author chose to utilise two main development environments each of which served specific purposes aligned with different aspects of the application's development.

##### Visual Studio Code (VS Code)

Visual Studio Code (VS code)<sup>1</sup> was chosen as the code editor for both frontend and backend development for its ability to support both React Native and Django. Moreover, VS code offers a small download (<200 MB) and has a disk footprint of less than 500 MB<sup>2</sup> which proved to be invaluable considering the plethora of files and repositories created throughout development and the need for fast, easy development. Finally, VS Code was an editor that the author was already familiar with, and had a preference for, and did not have to be downloaded as that had already been done previously.

##### Android Studio

The author utilised Android Studio<sup>3</sup> for its Android application development capabilities. It facilitated the management of virtual device emulation, as the software allows for testing React Native applications under various simulated devices. This allowed the author to test their application more thoroughly across different Android versions and screen sizes. Furthermore, Android Studio was linked with VS code, so after each compilation of the project there was no additional need to open it separately, which would have occupied valuable CPU space otherwise.

#### 4.1.2 Graphics and Diagram Creation

The graphical elements of the application - game character, backgrounds, accessories for the game character as well as some of the diagrams present in this paper were created using a combination of Paint<sup>4</sup> and Paint3D<sup>5</sup>. These tools were leveraged for their accessibility and ease of use, given the constrained project timeframe and the author's initial unfamiliarity with more complex design software. Utilising Paint and Paint 3D allowed for a straightforward design process without the steep learning curve associated with more advanced design tools. This approach ensured that the development of visual content

---

<sup>1</sup><https://code.visualstudio.com/>

<sup>2</sup><https://code.visualstudio.com/docs/supporting/requirements>

<sup>3</sup><https://developer.android.com/>

<sup>4</sup><https://apps.microsoft.com/detail/9pcf5b6t72h?hl=en-US&gl=US>

<sup>5</sup><https://apps.microsoft.com/detail/9nblggh5fv99?rtc=1&hl=en-gb&gl=GB>

remained efficient and effective, only requiring for the student's artistic capabilities on paper, which proved especially beneficial.

### 4.1.3 Database Management Software

For database management, pgAdmin<sup>6</sup> was employed to interact with the PostgreSQL database mentioned in Chapter 3- Technical Review and System Architecture. The pgAdmin software provided a user-friendly graphical interface to directly execute and manage SQL queries, facilitating a more intuitive interaction with the database, needed to perform manual queries when the author wanted to check the database without the help of Django's Object Relational Mapping functionality.

### 4.1.4 Platforms and Testing Devices

The development of the application was conducted on a Windows operating system (OS), which offers compatibility with VS code and Android Studio. However, iOS emulators are not available on Windows and the author had no access to a Mac Device. Therefore, for the time being, the author focused on Android implementation.

#### Device Testing

- **Emulators:** As mentioned above in section 4.1.1, Android Studio was used for emulators operating on Android. Managing the same for iOS was not possible as there are no options (or at least free of charge) available for Windows OS. Testing on an emulator was essential for a faster development, as it allowed for changes in code to be seen in real time with no need to rebuild the project. However, while emulators offer the convenience of quickly switching between device profiles, they cannot perfectly replicate the nuances of physical hardware interactions.
- **Physical Devices:** Testing the application on real devices played a critical role in assessing the application's performance and UI/UX design in a practical environment. The author used their Lenovo Chromebook as well as a Samsung phone for testing on Android. To build a '.apk' file, the author used Gradle, a build automation tool for multi-language software development, directly from the main repository of the project.

### 4.1.5 Version Control

To support the development of the application, a source code management tool was utilised to prevent data loss. The author employed Git and GitHub.

#### Git and GitHub

Git is a distributed version control system that tracks changes in any set of computer files, commonly used by programmers for collaboration and source code management. Git's objectives include its speed, data integrity and capability to support workflows in a non-linear, distributed way through branching and merging. GitHub is a developer platform that utilises Git and gives a platform on which developers can create, store, manage and share their code. Furthermore, this version control system provides developers with the capability to revert modifications made to their code.

#### Repository Management

The author maintained two primary repositories: one dedicated to backend operations and the other to frontend development. This allowed for easier and faster development with simpler code maintenance.

---

<sup>6</sup><https://www.pgadmin.org/>



## 4.2 Dependency Management

Managing dependencies was pivotal in maintaining the scalability, performance, and security of the application. Listed in the subsections below is a detailed overview of the dependencies for both the frontend and backend parts of the project. Firstly, the author will examine the frontend dependencies.

### 4.2.1 Frontend Dependencies

During the time of development, utilising the immense React Native ecosystem was pivotal to enhance the functionalities and development process of the application. The React Native framework provides developers with a plethora of pre-built libraries and components that can be employed instantly to offer a wide variety of features such as animations, music and even text-to-speech. Nevertheless, integrating third-party libraries can attend to introducing potential security vulnerabilities and maintenance issues. This is why the author thoroughly examined the selection of such with attention to several key factors such as:

- **Activity and Maintenance:** The libraries selected for the application were thoroughly examined. The author reviewed for their active maintenance: commit history on GitHub, frequency of updates, and monitoring the resolution of issues. As a consequence of that, the author was certain that the libraries they chose were reliable, being both up-to-date and actively patched for any known vulnerabilities.
- **Community Trust:** Libraries that are renowned within the developer community tend to be more frequently tested which could potentially help in identifying and resolving technical and security issues. Moreover, reviews by the community and ratings on platforms such as npm, GitHub and Stack Overflow provided the author with invaluable insights about the reliability and performance of the relevant libraries.
- **Project Compatibility:** The libraries mutual compatibility was also examined and tested by the author to ensure there would not be any potential future conflicts between any of the modules imported.

Having taken all of those factors into consideration, the author chose to utilise the following libraries:

- **react-native-community/blur<sup>7</sup>:** This library provides a component that blurs everything underneath it. It is commonly utilised for navigation bars, tab bars and modals. It is especially useful for creating a focus effect in the UI, that enhances users' focus and improves the overall aesthetic of the application. For this project, the author used Blur View when displaying modals on the screens to provide smoother transition and greater focus as normal modals seemed to blend with the content of the application.
- **react-navigation/material-bottom-tabs<sup>8</sup>:** Provides themed tab bar navigation on the bottom of the screen that lets the user switch between different screens with seamless animation. It allows for a familiar user experience and renders identical on both Android and iOS.

---

<sup>7</sup><https://github.com/Kureev/react-native-blur>

<sup>8</sup><https://reactnavigation.org/docs/material-bottom-tab-navigator/>

- **react-navigation/native<sup>9</sup> and react-navigation/native-stack<sup>10</sup>**: Native Navigation allows developers to create applications which have means to transition between screens seamlessly where each new screen is placed on top of a stack using *native-stack*. Those libraries are critical components in applications' architectures, enabling a fluid and intuitive navigation experience. Additionally, *react-navigation/native* and *react-navigation/native-stack* use native primitives - the core, low-level elements provided by a device's operating system, enabling for smoother animations and optimized performance for a consistent user experience.
- **axios<sup>11</sup>**: In the development of the application, the author chose to utilise *axios* to handle HTTP requests to the backend as it provides promise-based structure that simplifies asynchronous code. Asynchronous code allows a program to perform tasks concurrently without waiting for each task to complete before starting the next. This approach helps maintain responsiveness and efficiency, especially in operations that involve delays, in the case of this project-requests to the backend. Furthermore, *axios* stands out with features like interceptors for modifying requests and responses, automatic JSON data transformations (lightweight format to transmit data over the Internet). While alternatives such as the native *fetch API*<sup>12</sup>, *superagent*<sup>13</sup>, and traditional *XMLHttpRequests*<sup>14</sup> were considered by the author, *axios* provided a more robust and developer-friendly toolkit. For instance, *Fetch*, does not have built-in support for interceptors and is more verbose in error handling, whereas *superagent*, although lightweight, does not natively support promises as efficiently as *axios*.
- **lottie-react-native<sup>15</sup>**: A library that enables programs to use animations by rendering After Effects animations in real time. Additionally, *Lottie* provides an abundance of free animations, one of which was utilised by the author to congratulate a user on brushing session completion.
- **react-native-reanimated<sup>16</sup>**: An animation library that has the ability to render animations natively on mobile devices, improving the performance of UI transitions. In the context of the author's project, *Reanimated* was utilised to provide smooth transitions between the Log In and Sign Up screens in addition to displaying accessories for the character of the application.
- **react-native-calendars<sup>17</sup>**: The React Native Calendar package, which provides uniform aesthetics for both iOS and Android, includes various customisable calendar components. The author utilised it for the Parent Section of the application, displaying each user's active days.
- **react-native-sound<sup>18</sup>**: React Native module for playing sound clips on iOS, Android, and Windows. It provides pausing and resuming functionality as well as playing on loop, which was pivotal for the brushing functionality of the application and utilised by the author to implement a background and a toothbrushing sound.

---

<sup>9</sup><https://reactnavigation.org/docs/getting-started/>

<sup>10</sup><https://reactnavigation.org/docs/native-stack-navigator/>

<sup>11</sup><https://axios-http.com/docs/intro>

<sup>12</sup>[https://developer.mozilla.org/en-US/docs/Web/API/Fetch\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API)

<sup>13</sup><https://www.npmjs.com/package/superagent>

<sup>14</sup><https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest>

<sup>15</sup><https://lottiefiles.com/free-animations/react-native>

<sup>16</sup><https://docs.swmansion.com/react-native-reanimated/>

<sup>17</sup><https://www.npmjs.com/package/react-native-calendars/v/1.1286.0>

<sup>18</sup><https://www.npmjs.com/package/react-native-sound>

- **react-native-animated**<sup>19</sup>: This library is "designed to make animations fluid, powerful, and painless to build and maintain". The author leveraged that fact to implement the tooth brushing functionality of the application (positioning the toothbrush on different places on the screen and then using a range of inputs and outputs for rendering a moving animation) and for the progress screen (the dog moving up on a path once again accomplished using the same technique).
- **react-native-tts**<sup>20</sup>: This library converts text into speech within the application. It was utilised for the main functionality of the project, reading out loud instructions to the user, to promote adhering to adequate brushing techniques. It allowed the author to modify the speech rate, pitch, volume and language in order to achieve a 'child-like', friendly voice, sounding less 'robotic'.
- **react-native-vector-icons**<sup>21</sup>: A library that contains a plethora of free-to-use icons which were necessary to create a user interface that is visually appealing. For their application, the author employed the *Material*<sup>22</sup> icon set which was used in the bottom tabs navigation bar.
- **react-native-encrypted-storage**<sup>23</sup>: Security is often overlooked by developers when creating applications. However, it is one of the most critical ones as often applications will store sensitive information about its users. One solution to storing data is *Async Storage*<sup>24</sup> which is a community maintained library for React Native that provides an asynchronous, unencrypted storage. However, *Async Storage* does not provide means to store sensitive user data. React Native's *Encrypted Storage* is one solution to this issue that the author opted for in their application. It works by acting as a wrapper for Android's *EncryptedSharedPreferences*<sup>25</sup> and iOS' *Keychain*<sup>26</sup>.
- **zustand**<sup>27</sup>: Managing states (variables' values) across multiple files can become complex and verbose when applications scale up. The author put in place *Zustand*, a global state management system, to solve this. Utilising this library makes managing states simpler and reduces the need for prop drilling - inheritance of variables from parent to child components. This approach helped the author simplify the structure of their application and sped up the development process once the library was set up. *Redux*<sup>28</sup> is an alternative to Zustand, but it may be very verbose and sophisticated, and it doesn't offer an easy-to-use method of managing states that reduces boilerplate code and speeds up development.

## 4.2.2 Backend Dependencies

Backend development is essential for managing data and security. Additionally, it makes the frontend more responsive and lightweight as a significant amount of the processing operations such as data retrieval and storage can be offloaded to the backend. In this way, the frontend can remain focused on delivering a fast, interactive and smooth experience to the user. Applications may easily scale up using

---

<sup>19</sup><https://reactnative.dev/docs/animated>

<sup>20</sup><https://www.npmjs.com/package/react-native-tts>

<sup>21</sup><https://oblador.github.io/react-native-vector-icons/>

<sup>22</sup><https://static.enapter.com/rn/icons/material-community.html>

<sup>23</sup><https://www.npmjs.com/package/react-native-encrypted-storage>

<sup>24</sup><https://reactnative.dev/docs/security>

<sup>25</sup><https://developer.android.com/reference/androidx/security/crypto/EncryptedSharedPreferences>

<sup>26</sup>[https://developer.apple.com/documentation/security/keychain\\_services/](https://developer.apple.com/documentation/security/keychain_services/)

<sup>27</sup><https://github.com/pmndrs/zustand>

<sup>28</sup><https://redux.js.org/>

this strategy as the amount of data and processes that need to be handled increases. The choice was made to make use of the Django REST framework, as was indicated in Chapter 3 - Technical Review and System Architecture. Below is a detailed exploration of each backend dependency that came with the Django and the creation of a secure backend. Below are listed some of most important dependencies that come with DRF as well as some dependencies that the author utilised for code readability:

- **django-cors-headers**<sup>29</sup>: This library is vital for handling Cross-Origin Resource Sharing (CORS). It allows the React Native frontend application, hosted on a Heroku server, to safely interact with the backend by setting appropriate HTTP headers that permit cross-domain requests.
- **django-environ**<sup>30</sup>: This tool is used for managing environment variables. It helps configuring the application with secrets and settings stored in environment variables, thus keeping sensitive information out of the source code as mentioned in Chapter 3 - Technical Review and System Architecture and as depicted in Figure 3.6 contained in that chapter that shows an example of how environment variables may be used.
- **gunicorn**<sup>31</sup>: "Green Unicorn" - Gunicorn is a Python Web Server Gateway Interface (WSGI) HTTP server needed as a dependency to establish communication between the Heroku server and Django REST backend.
- **django-rest-framework-simplejwt**<sup>32</sup>: As mentioned in Chapter 3 - Technical Review and System Architecture, the author's backend utilises the Django REST JWT tokens to complement the application's security. This package adds JSON Web Token (JWT) authentication capabilities to Django REST Framework, providing a secure and scalable way to manage user authentication and session management without storing session states on the server.
- **psycopg2-binary**<sup>33</sup>: This PostgreSQL adapter is the most popular PostgreSQL database adapter for the Python programming language. It facilitates communication between the Python application and the PostgreSQL database, allowing database queries and operations to be conducted through Python - also known as Object Relational Mapping.
- **whitenoise**<sup>34</sup>: Whitenoise is used to efficiently serve static files directly from Django in a production setting. Static files are all of the files that are part of the application's code but do not change dynamically during runtime. Whitenoise integrates seamlessly with Django to manage and cache these files, eliminating the need for a separate web server for these resources, making everything simpler and faster.
- **black**<sup>35</sup>: This formatting tool ensured that the author's Python codebase followed a consistent style by reformatting the code to meet styling conventions, improving readability and maintainability, aiming to adhere to PEP 8<sup>36</sup>, the Python style guide.

---

<sup>29</sup><https://pypi.org/project/django-cors-headers/>

<sup>30</sup><https://django-environ.readthedocs.io/en/latest/>

<sup>31</sup><https://gunicorn.org/>

<sup>32</sup><https://django-rest-framework-simplejwt.readthedocs.io/en/latest/>

<sup>33</sup><https://pypi.org/project/psycopg2-binary/>

<sup>34</sup><https://whitenoise.readthedocs.io/en/stable/django.html>

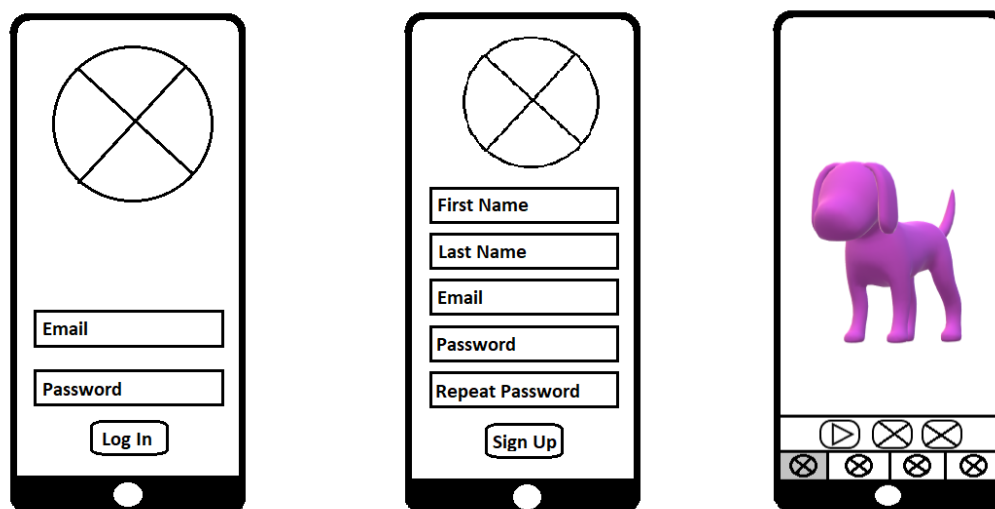
<sup>35</sup><https://github.com/psf/black>

<sup>36</sup><https://peps.python.org/pep-0008/>

### 4.3 App workflow and Wireframes

In the following subsections the author will give a detailed explanation of each screen contained within their application and their intended use. The description will commence with the Log In, Sign Up and Home screens

#### 4.3.1 Log In, Sign Up and Home screens



**Figure 4.1:** This figure depicts a wireframe of the first 3 screens in the author's application - Log In, Sign Up and Home, shown left to right. It demonstrates an approximate outline of the components on each screen

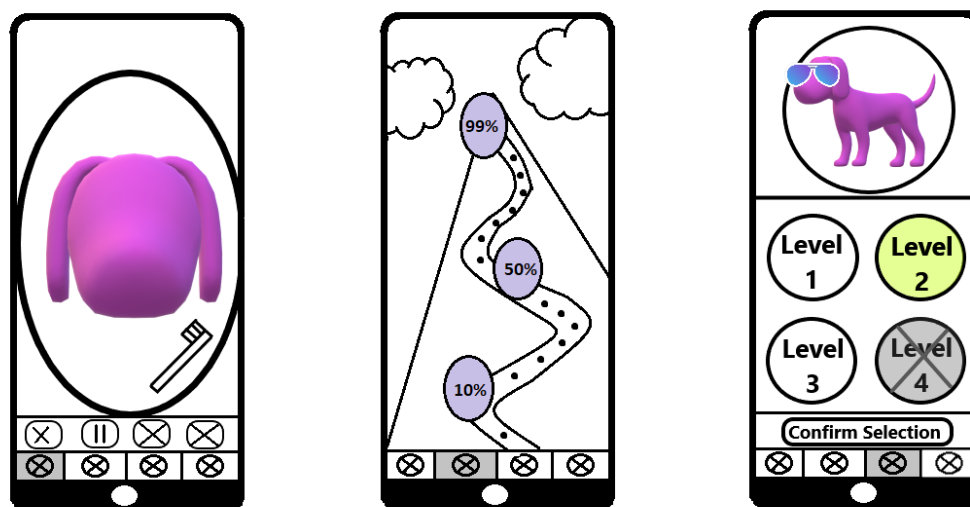
The Log In , Sign Up and Home screen wireframes can be seen on Figure 4.1 (left to right in order). The figure demonstrates an approximate outline of the components on each screen. The author will now provide a detailed explanation of each of them in order. The application begins with two primary entry points for users: the Log In and Sign Up screens. These terms were chosen as they adhere to standard naming conventions<sup>37</sup>, making them immediately recognisable and intuitive for new users. Both of those screens include text input fields for the required credentials to either create an account or sign into one. Additionally, they feature input validation to ensure data integrity and enhance user experience. For instance, if a user fails to enter an email or enters one in the incorrect format, the input box's corners would turn into red colour, and an appropriate error message is displayed (for instance "Please enter a valid email"). Similarly, to enhance the security of the application, if there is a mismatch in the credentials, email and password, during login, instead of specifying which one is incorrect, the application displays a generic "Invalid credentials" error message. This approach helps in preventing potential security breaches by not revealing which part of the authentication data might have been correct. To further improve security, the input validation also requires that users include at least 1 lowercase and 1 uppercase letter, 1 special character, and be at least of length 8, making each password less prone to attacks (Yıldırım and Mackie, 2019). The required fields for user creation include email, first and last name and a password. The password needs to be repeated to ensure that the user is aware of the exact

<sup>37</sup><https://learningenglish.voanews.com/a/register-sign-in-and-log-in/6333897.html>

password they have chosen, reducing the risk of accidental mistypes that could lead to login issues later on. Additionally, instead of a username as the unique identifier for each user, an email was utilised to allow a future application functionality of resetting a forgotten password and email verification.

Once logged in, the user lands on the Home screen, which features a dog character situated in a bathroom setting. This character is central to the application's interactive experience and aims to capture the benefits of utilising a game character mentioned in Section 2.5.4. The screen displays the user's current level and includes a 'Start' button to initiate a brushing session.

### 4.3.2 Home, Progress and Shop Screens



**Figure 4.2:** This figure depicts a wireframe of the second 3 screens in the author's application - Home, Progress and Shop, shown left to right. It demonstrates an approximate outline of the components on each screen

The Home, Progress and Shop screen wireframes can be seen on Figure 4.2 (left to right in order). The figure demonstrates an approximate outline of the components on each screen. The author will now provide a detailed explanation of each of them in order. As mentioned above, the Home Screen features a 'Start' button. This button is used to commence a tooth brushing session. Upon starting, the user zooms into an image of the dog with human teeth instead of dog teeth. This design choice was made to better demonstrate proper brushing techniques, as human teeth layouts are more relatable and educational for the intended audience. During the brushing session, the application leverages the React Native Animated library to animate the toothbrush movements around the dog's mouth, illustrating adequate brushing techniques detailed in Section 2.2. This session is accompanied by a background sound and toothbrush noises sourced from a free website - Uppbeat.io<sup>38</sup> that provides royalty-free music for content creators. While they do offer some free music tracks. The author used the *Play Time* sound by Andrey Rossi. It was carefully chosen to complement the interactive experience without detracting from the educational purpose of the application complementing to the Aesthetics in the application as

<sup>38</sup><https://uppbeat.io/>

mentioned in section 2.5.4 about the Tetrad Framework. Additionally, using React Native's Text to Speech functionality, the dog provides simple verbal instructions, helping guide the child through the brushing process. When a user logs in for the first time, they are prompted to name their character, personalising the experience and enhancing engagement by having the dog refer to itself by the chosen name during instructions (specifically - "Hi, it's your furry friend ...").

The second screen features an interactive progress bar visualized as a mountain. This mountain represents the user's progress through levels, relative to the experience points required to advance. The user's position on the mountain correlates with their accumulated points, as suggested by the middle image in Figure 4.2, visually encouraging progress and continued interaction with the application. For instance, if a user has 0 points out of 120, the character would be displayed at the bottom of the mountain. Conversely, if they had 60, the dog would be positioned in the middle of the path. Additionally, on screen rendering, the dog moving up to the appropriate position on the mountain is animate to enhance the interactive element of the application. The experience points are based on the total seconds a user has brushed their teeth. To encourage initial motivation, the required seconds to pass the first level a 120 - the time required to complete one brushing session. To calculate the maximum experience points (XP) needed for each subsequent level, the following formula was created by the author:

$$\text{increase\_factor} = 1.2$$

$$\text{new\_level} = \lceil \text{increase\_factor} \times \text{userDetails.current\_level\_max\_xp} \rceil$$

$$\text{setMaxLevelXP}(\{\text{newMaxLevelXP} : \text{new\_level}\})$$

In this formula:

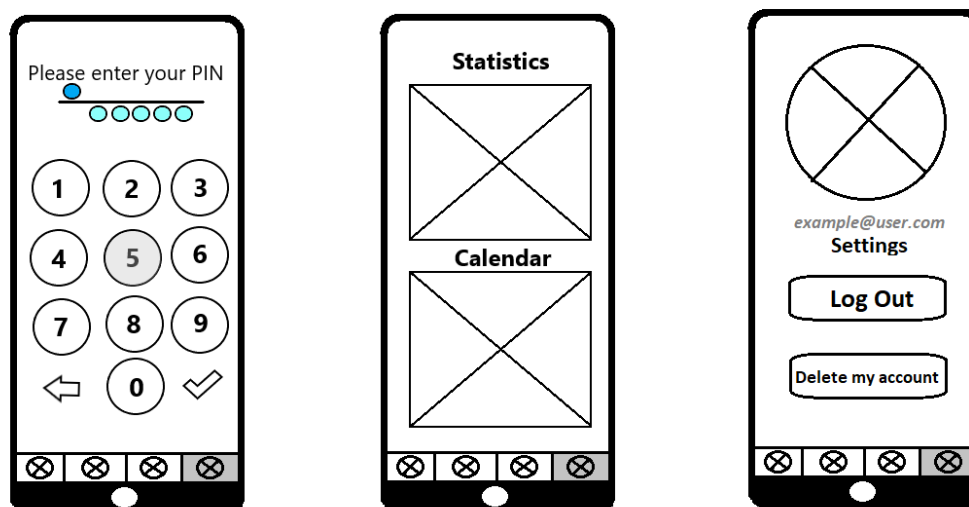
- **increase\_factor:** A factor by which the current level's maximum XP is incremented. In this case, it's set to 1.2, indicating an increase of 20%. Which provides a balanced progression system through the levels.
- **new\_level:** This variable calculates the new level based on the current level's maximum XP multiplied by the increase factor. The `Math.ceil()` function rounds up the result to the nearest integer, as levels are typically represented as whole numbers.
- **setMaxLevelXP(newMaxLevelXP: new\_level):** This function sets the new maximum level XP using the Zustand<sup>39</sup> global state manager with a property `newMaxLevelXP` and its corresponding value `new_level`.

Finally, the third screen is a mini-shop where users can unlock accessories for their dog character as they complete levels. For each level they complete, a new accessory is unlocked, allowing users to customise their character. These accessories remain with the character even after logging out, providing a continuous personalised experience across sessions. Currently, the application has accessories implemented for 10 levels.

---

<sup>39</sup><https://zustand-demo.pmnd.rs/>

### 4.3.3 Parents, Statistics and Settings Screens



**Figure 4.3:** This figure depicts a wireframe of the first 3 screens in the author’s application - Parents, Statistics and Settings, shown left to right. It demonstrates an approximate outline of the components on each screen

The Parents, Statistics and Settings screen wireframes can be seen on Figure 4.3 (left to right in order). The figure demonstrates an approximate outline of the components on each screen. The parent component, protected by a six-digit PIN set during the first use of the application (with reset options available via email and password). This section allows parents to access settings to log out or delete their account and view detailed statistics about their child’s brushing habits. These statistics include brushing streaks, total number of brushes, and completion percentages for morning and evening sessions, alongside a calendar that displays badges for completed sessions—full badges for both morning and evening brushes, and partial badges for one-session days.

## 4.4 Communication Between Components: System Architecture Diagram

This section describes the communication between the frontend and backend. As outlined in Chapter 3 - Technical Review and System Architecture, the frontend, implemented using React Native, and the backend, powered by Django REST, establish their communication through a network of HTTP requests and responses. The author now presents a high-level overview of the relationship between the frontend screens and backend Views, briefly mentioned in that chapter. Figure ?? offers a visual representation of each View and screen, with certain details omitted for simplicity.

The main components illustrated in the diagram are the React Native Navigation, the React Native Frontend, and the Django REST Views. Navigation between screens is facilitated by three main sub-components: App.js, TabNavigation.js, and ParentsNavigation.js. App.js establishes navigation between the Log In, Sign Up, and Home screens. TabNavigation.js then provides a bottom tab view that allows users to navigate between the Home, Progress, Shop, and Parents screens. Lastly, the Parents screen employs its own ParentsNavigation.js, which connects the PIN screen with the Statistics and Settings



screens.

Whenever these screens need to fetch or send data to the backend server, they do so through the URL endpoints depicted in grey in the diagram. For example, to sign in from the Log In screen, the frontend sends a request to the '/signin/' endpoint. Upon successful authentication, the backend sends an HTTP\_200\_OK response back to the Home screen. The React Native Navigation in App.js then directs the user to the Home page.

## 4.5 Design Choices

As mentioned in Section 2.5.4, the Mechanics, Story, Aesthetics, and Technology play a crucial role in the overall design of a game. In this section, the author will discuss some of the most critical design choices they took during the creation of their application in regards to Aesthetics.

### 4.5.1 Naming the Application

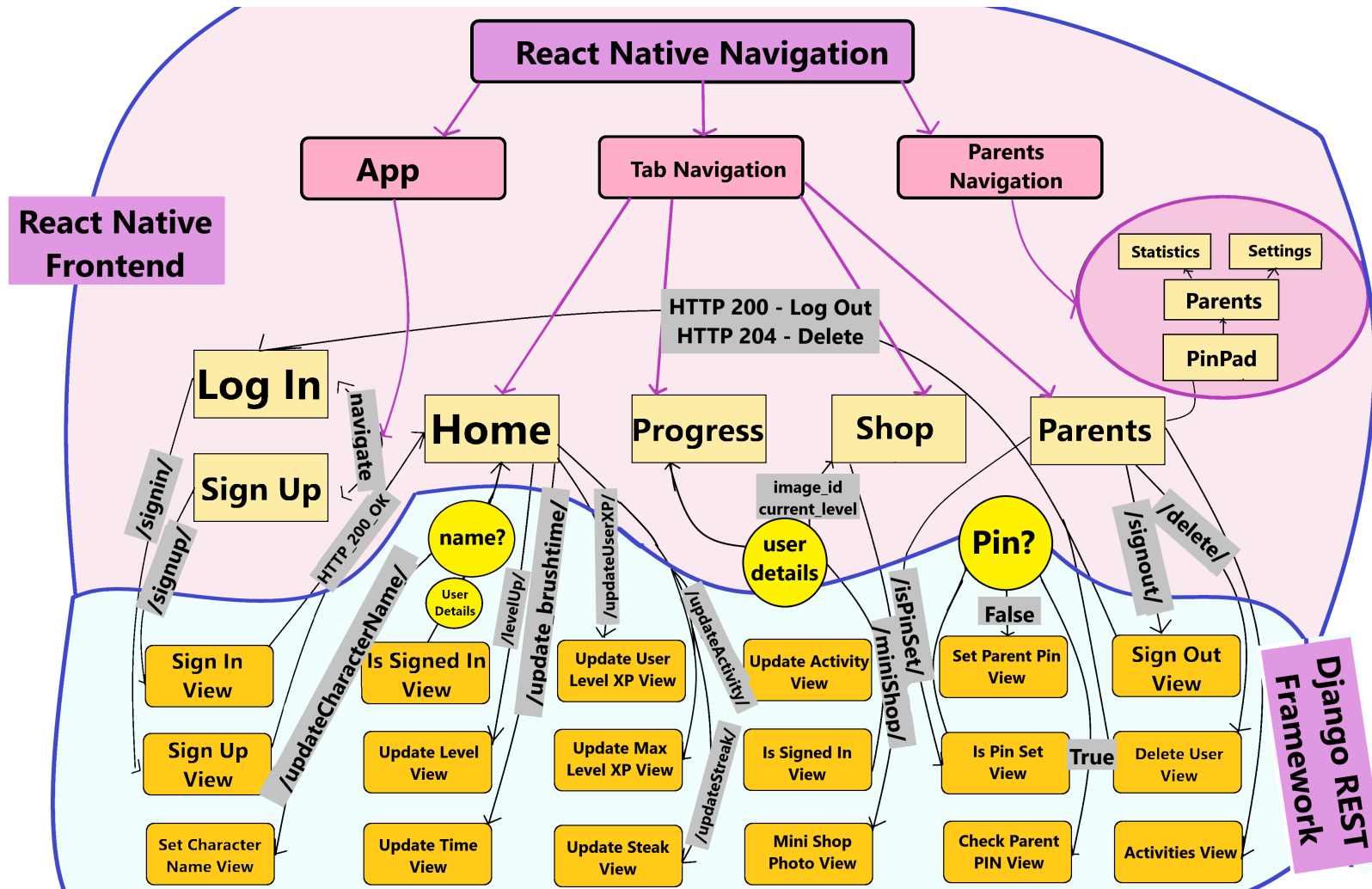
As with every application, the author had to pick a name for theirs. In collaboration with the application's client, the final choice to name the application 'Brushy' was taken. This was due to the reason that both the student and the client wanted the name to encapsulate the application's purpose while remaining catchy and child-friendly. Additionally, the name had to be intuitive for children, suggesting the app's focus on brushing in a playful and memorable manner.

### 4.5.2 Graphics and Demographic Considerations

To design 'Brushy', the graphics and design elements were chosen to be suitable for the developmental stages (Deater-Deckard et al., 2013) of the target demographic—children aged 4 to 6 years. This age group was inspired by Jesse Schell's description in "The Art of Game Design,". According to him, children at these ages are just beginning to show interest in games, choosing simple game play (Jesse Schell, 2008, Ch. 8, p. 99-101). 'Brushy' incorporates these principles by leveraging a central game character (Lim et al., 2009). The character-led approach helps to abstract the learning process into a friendly and approachable game (Ketamo, 2015; Akram, 2021), making it less of a chore and more of an engaging activity. The insights provided by Dr. Rachelle Ho<sup>40</sup> from the School of Psychology at the University of Aberdeen during a meeting with the author about children's developmental stages were crucial. Dr. Ho emphasised that the author's idea of utilising a game character and simple rewards could significantly motivate children in these early stages. She additionally said that a variety of colours would be stimulating for children, guiding the student to create a vibrant and visually appealing and engaging aesthetics within the application. The choice of blue as the primary colour for the application was taken due to the fact that blue has an universal appeal and it is associated with calmness and cleanliness—attributes that aligned well with the application's oral hygiene focus (Yang and Shen, 2022; Bonnardel et al., 2011). Additionally, blue is often favoured across genders (Ellis and Ficek, 2001), making it an inclusive choice for a diverse user base.

---

<sup>40</sup><https://www.abdn.ac.uk/psychology/people/profiles/rachelle.ho>



**Figure 4.4:** This figure depicts a wireframe of the second 3 screens in the author’s application - Home, Progress and Shop, shown left to right. It demonstrates an approximate outline of the components on each screen

## Chapter 5

# Evaluation

### 5.1 Introduction

To evaluate the current usability and collect user feedback, the author conducted a small User Evaluation Survey. This chapter focuses on the methodology used and the key results founds after statistical analysis. The responses and survey questions can be found the the AppendicesA.

### 5.2 Methodology

The survey consisted of both quantitative and qualitative questions, including the System Usability Scale<sup>1</sup> (SUS), adapted for children, to measure usability and learnability quantitatively. Other questions were designed to assess users' initial reactions, the ease of application navigation, character engagement and whether rewards were effective. The author constructed their survey using Google Forms<sup>2</sup> and distributed it to parents with children, who have Android devices to test the application before filling out the survey.

### 5.3 Statistical Methods Employed

To analyse the data collected from the survey, the following statistical methods were employed:

- **Descriptive Statistics<sup>3</sup>:** Used to provide a summary of the SUS scores and responses to other quantitative questions.
- **Frequency Distribution<sup>4</sup>:** Analysed to determine the distribution of responses related to character engagement and the application's features.
- **Mean:** Calculated for overall usability scores.

#### User Engagement and 'Brushy' Features

- **Character Engagement:** Frequency counts of different levels of engagement reported by users.
- **Most Enjoyed Features:** Frequency counts detailing which features users found most appealing.

#### Behavioral Changes

- **Knowledge Gained:** Number of users acknowledging new information learned through the app.

---

<sup>1</sup>[https://en.wikipedia.org/wiki/System\\_usability\\_scale](https://en.wikipedia.org/wiki/System_usability_scale)

<sup>2</sup><https://www.google.co.uk/forms/about/>

<sup>3</sup>[https://www.investopedia.com/terms/d/descriptive\\_statistics.asp](https://www.investopedia.com/terms/d/descriptive_statistics.asp)

<sup>4</sup><https://www.scribbr.com/statistics/frequency-distributions/>

## 5.4 Results

### 5.4.1 SUS Results

To calculate the adjusted scores for each SUS question, the author did the following:

- For odd-numbered questions (positive questions): Subtract 1 from the response.
- For even-numbered questions (negative questions): Subtract the response from 5.
- Calculated the mean and standard deviation for each question based on the adjusted scores. Aggregated the SUS scores to compute the overall mean and standard deviation for the entire SUS questionnaire.

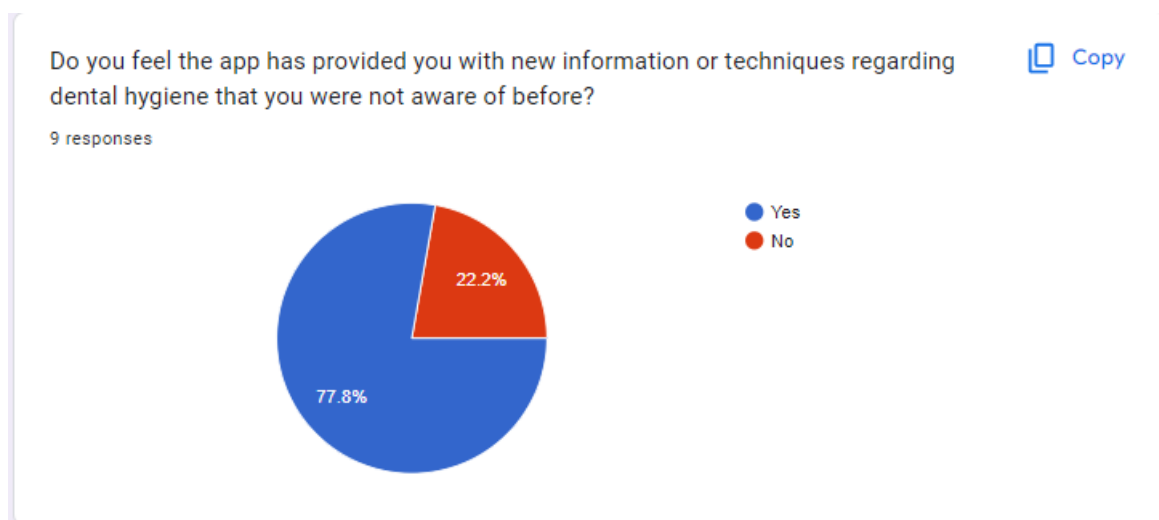
Based on the adjusted System Usability Scale (SUS) calculations, the data revealed the following results:

- **Mean SUS Score:** The average adjusted SUS score was calculated to be 72.5.

### 5.4.2 Descriptive Statistics

The author will now list some of the descriptive statistics they collected from their survey. Their pie charts can be found in the Appendices A.

- **Ease of Navigation:** The average score was 3.88 out of 5.
- **Character Engagement:** The average score was 4 out of 5.
- **Knowledge Gained:** 77.8% of the parents acknowledged gaining new knowledge about dental hygiene from using the application as can be seen in Figure 5.1.



**Figure 5.1:** In the survey conducted, 77.8% of the parents acknowledged gaining new knowledge about dental hygiene from using the application while 22.2% said they did not

- **Parental Supervision:** The average score was 3.67 out of 5

### 5.4.3 Frequency Distribution:

#### Character Engagement Levels:

- Very Engaging: 44.4%

- Somewhat Engaging: 22.2%
- Neutral: 22.2%

**Most Enjoyed Features:**

- Dressing up the character and leveling up: 44.4%.
- Earning rewards or achievements: 22.2%.
- The interactive brushing activities: 15%

**5.4.4 Behavioral Changes****Adherence to the brushing techniques in the application**

- **Always:** 44.4%
- **Sometimes:** 44.4%
- **Rarely:** 11.1%

**5.5 Conclusions**

The author will discuss the conclusions drawn from this evaluation in Chapter 6 - Results.

## Chapter 6

# Results

The author will now discuss the results of the survey in the previous chapter.

### 6.1 SUS Results

The mean System Usability Scale (SUS) score was 72.5, which is above average, indicating that users found the app to be relatively usable. That score translates to good usability<sup>1</sup> (second after 'Excellent') according to industry standards. However, the SUS score alone does not account for all aspects of user experience such as engagement and educational impact.

### 6.2 Descriptive Statistics

The average scores for various components of the application like ease of navigation (3.88 out of 5), character engagement (4 out of 5), and parental supervision (3.67 out of 5) were considered. These scores suggest that users generally found the application easy to navigate and engaging. The score for parental supervision indicates a moderate reduction in the need for parental oversight, which aligns with the application's objectives.

### 6.3 Frequency Distribution

Notably, the high engagement with character features (44.4% found it very engaging) suggests that the gamification aspects were effective in holding children's attention. Additionally, 44.4% of the participants indicated that the most enjoyable feature of the application was dressing up the character and leveling up suggesting that the relationship between those two components was well integrated. Finally, the author's primary objective was to teach adequate brushing techniques, and as indicated by participants, 44.4% of children adhered to the techniques shown in the application, with 44.4% of the rest of the group also selecting sometimes as an answer, indicating a positive reinforcement.

### 6.4 Behavioral Changes

The adherence to brushing techniques demonstrated varied commitment, with an equal percentage of parents (44.4%) reporting that their children always or sometimes followed the app's techniques, which suggests partial success in changing behaviors.

### 6.5 Limitations of the Study

Several limitations arise from the methodology and external factors of the author's study:

---

<sup>1</sup><https://uiuxtrend.com/measuring-system-usability-scale-sus/>

### Small Sample Size

With only 9 participants, the study lacks statistical power, which makes it difficult to generalise the findings to a larger population. This small sample size also increases the margin of error and variability in the data.

#### **6.5.1 Selection Bias**

The sample consisted only of parents with Android devices and children aged 4-6 willing to participate. This narrows the demographic and technological representation of the study, potentially excluding insights from users of other age groups that could potentially also use the application.

#### **6.5.2 Self-reported Data**

The results rely on honesty in parental reporting, which can introduce bias. Additionally, parents were required to answer on behalf of their children, which might not capture the exact opinions of children.

#### **6.5.3 Lack of Diverse Demographic Data**

Not taking into account the participants' backgrounds or personal information, which was the case for the author's evaluation, could overlook significant factors that influence the effectiveness of the application. For example, socioeconomic status or previous exposure to dental hygiene education might affect the results.

These limitations suggest that while the initial findings are very promising, further research with a more robust methodology and a larger, more diverse sample size is necessary to validate the results and refine the application.

## Chapter 7

# Discussion

### 7.1 Project Summary

This project presented the development and evaluation of a client-based application designed to enhance pediatric dental hygiene through gamification strategies.

The project was conceptualised by observing the daily struggles faced by the National Health Service and the widespread consequences of poor dental habits among children within the UK. To address this, the author developed an Android mobile application utilising React Native and the Django REST Frameworks, ensuring a scalable and secure application environment. The application featured a game character to guide children through the brushing process, a reward system to enhance engagement, and a parental interface for monitoring progress. The primary goal was to create an engaging, educational tool that could teach proper dental hygiene habits among children while minimising the need for parental supervision.

Throughout the development cycle, the author followed the Agile methodologies, ensuring the timely delivery of the application. User evaluations conducted post-implementation highlighted the application's success in enhancing engagement and improving dental hygiene habits among children.

However, not all objectives were met without challenges. Certain technical and operational issues were encountered, which provided learning opportunities and insights for future implementations.

In the next two sections the author will speak about what worked well and what did not during the duration of this project.

### 7.2 What worked well

#### 7.2.1 For The Author

During the duration of this project, the author experienced significant personal growth and development, particularly in the ability to deliver a fully functional application, created solely by them. They gained a comprehensive understanding of full-stack technologies which improved their technical skills significantly.

Time management was another critical skill that the author had to learn during this project. The thesis allowed the student to learn how to effectively allocate time to different stages of the development process, from initial research to final testing and regular client meetings.

Working in collaboration with a client was also a new experience for the author. This interaction provided real-world insights into the dynamics of client-developer relationships, emphasising the importance of communication and understanding client needs to deliver a product that meets their expectations.



### 7.2.2 Application-wise

From a technical perspective, the author managed to create a fully functional application that is nearly ready for release on Google Play. The seamless integration of various components—frontend, backend, and database management—ensures that the application operates efficiently and securely.

The risk management strategies implemented were largely successful, with all but one identified risk effectively mitigated. This success in risk avoidance underscores the efficacy of the preemptive measures taken, based on a thorough initial risk assessment undertaken in the beginning of the project.

## 7.3 Challenges

Despite the successes, there were several limitations that impacted the author's project. The use of Android Studio, while effective, proved to be resource-intensive, often consuming considerable CPU space and occasionally causing the student's laptop to crash. This technical challenge sometimes hindered the workflow and affected the author's productivity. To restart Android Studio and the emulator, the author needed at least 10 minutes at a time, sometimes more than half an hour.

Additionally, the design process of the graphics for the application proved to be very time-consuming. Although the author had excellent drawing experience on paper, the lack of knowledge in how to use design software other than Paint and Paint 3D proved to be one of the reasons for a slightly delayed product delivery date.

Adherence to the planned Gantt chart mentioned in Section 3.4 was another issue. Deviations from the scheduled milestones, particularly in the later stages of the project, led to a delay in the writing phase of this paper. The initial focus on the literature review, rather than an immediate dive into learning the necessary frameworks, contributed to a slower application development process. This misalignment in time management highlights a technical skills gap, mentioned in the Risk Assessment that was initially underestimated.

Additionally, time constraints prevented the implementation of some of the secondary objectives. These elements, while not critical to the core functionality of the application, could have enhanced the application and provided better user experience.

In the next section, the author will discuss all of the future implementations that have been planned.

## 7.4 Future Implementations

The future development of *Brushy* involves several innovative features aimed at enhancing user interaction, verifying engagement, and broadening educational and social aspects. These implementations focus on increasing the educational impact and user compliance through interactive and technical advancements.

### 7.4.1 Interactive Character Development

One significant enhancement involves the development of the application's gamified character. To encourage regular brushing habits, the character would be able to exhibit sadness when the user has not brushed their teeth for a specified period. This emotional response from the character aims to encourage children to brush their teeth by creating a sense of responsibility towards the character's happiness.

### 7.4.2 Verification of Brushing Activity

To ensure that children are actually brushing their teeth, a multi-stage development verification process is planned:

### **Parental Confirmation**

Initially, the application will include a functionality requiring parents to confirm via a PIN that their child has brushed their teeth. This stage enforces accountability and ensures that brushing routines are followed.

### **Facial Recognition**

The second stage involves implementing facial recognition technology to detect whether the user is in front of the camera during the brushing session, adding a layer of automated verification.

### **Advanced Image Recognition**

The final stage will use a Convolutional Neural Network<sup>1</sup> (CNN) to create filters that recognise the child's location, mouth, teeth, and toothbrush. This technology will provide real-time feedback and guidance during brushing, enhancing the educational value of the experience.

#### **7.4.3 Design and Accessibility Enhancements**

Based on more research and evaluation, the application design will be made more child-friendly and accessible. The interface will be simplified and made more intuitive to accommodate young users' cognitive abilities. General accessibility improvements will also be implemented to ensure the application is usable by children with varying abilities.

#### **7.4.4 Parental Educational Tools**

A new feature within the parental interface will provide educational content about oral hygiene. This resource aims to give parents knowledge and tips that they can pass on to their children, reinforcing good dental hygiene practices at home.

#### **7.4.5 User Engagement and Security Features**

##### **Push Notifications**

To improve regular usage, push notifications will remind users to brush their teeth, enhancing habit formation.

##### **Email Verification and Password Recovery**

Enhancements in user account management will include email verification for account registration and a password recovery system, improving security and user experience.

##### **Social Features**

Users will have the option to add friends within the application and either participate in simultaneous brushing sessions or send congratulatory "high fives" after completing brushing routines.

##### **Enhanced Password Security**

The application will include a feature to check for common passwords and prevent their use during account creation, increasing account security.

##### **Customization and Interaction**

- **Accessories and Tools:** Additional accessories for the character and options to change the toothbrush will be added to increase personalisation and engagement.
- **Interactive Character:** The game character will become more interactive, responding to touch with sounds and movements, making the application more engaging and enjoyable for children.

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)

- **Multiple characters:** For families with more than one child, the application will allow parents to have a character within the same account for each child. This feature includes the ability to track progress, customise settings, and receive notifications for each child individually.

These future implementations are designed to enhance the effectiveness of the application in promoting dental hygiene among children while also making the experience more engaging, educational, and user-friendly.

## **7.5 What I Would Do Differently If I Had To Start All Over Again And What I Have Learnt During This Project**

Now, I would like to switch up the tone of this paper and introduce the reader to a more personal subsection describing what I would change if I had to start all over again and what I have learnt during the duration of this project. To begin with, I will start with what I have learnt. During my university years, I have been exposed to multiple theoretical and programming courses. Through all of them, I have learnt many of the foundations of Computing Science. However, when it came to very big projects, I had never done one in my own time and had never been required to. This thesis project was a significant step in my journey towards improving my technical, theoretical and communication skills. I had some exposure to Software Engineering during my third year at university and from that, I learnt the importance of group collaboration and communication. Additionally, I learnt that I have a deep passion for Software Engineering and creating software. When I initially chose to do this topic, I met with Plamena Mya, the client of this application and I was touched by her dedication to a cause that would contribute to the greater good of the community. She motivated me to do well in this project and encouraged me regularly.

I have never been an organised person, and concentrating on multiple tasks at once has never been my strength. This is something I experienced throughout this project - I fell slightly behind a few times or set expectations that were too high for me to achieve in one semester. Going forward with the project, I can say I have developed something I am proud of and that I have almost learnt how to deliver mobile applications. If I had to start this project all over again, I would try to be more organised and less harsh on myself, setting realistic, manageable goals. This project not only challenged my capabilities but also significantly shaped my understanding of what it means to be a software engineer in practice, teaching me to balance ambition with attainable goals, and leaving me eager to apply these invaluable lessons in my future endeavors

# **Appendices**

## **Appendix A**

# **User Evaluation Survey**

This section provides all of the questions included in the author's user evaluation survey as well as summary of what each page of the survey was informing the participants about.

### **1. Introduction**

- Overview of the survey's purpose: Evaluating the usability, educational value, and engagement of a children's mobile application.
- Participants: Main users (children) and their parents/guardians.
- Process: Install the application, interact under supervision, and complete the survey using the System Usability Scale (SUS).

### **2. Important Information**

- Anonymity and confidentiality are ensured.
- Participation is voluntary with the right to withdraw at any time.

### **3. Consent Form**

- Agreement to terms and conditions is required to proceed.

### **4. Installation Instructions**

- Steps to install the mobile app, including downloading the Expo Go application and scanning a QR code or installing a .apk file.

### **5. Guided Tasks**

- Instructions for both children and parents to engage with the app.
- Tasks include finding the pet character, customising it, completing a brushing session, and accessing the parental dashboard.

### **6. Survey Questions-for children**

- a. What was your child's initial reaction to the app?
  - Very disinterested
  - Somewhat disinterested
  - Neutral

- Somewhat interested
- Very interested

b. How easy was it for your child to navigate the app without assistance?

- Very difficult
- Somewhat difficult
- Neutral
- Somewhat easy
- Very easy

How engaging was the character design for the child?

- Very disengaging
- Somewhat disengaging
- Neutral
- Somewhat engaging
- Very engaging

c. Which of the following features would you find most helpful in a mobile app for teaching kids to brush their teeth? (Select all that apply)

- Interactive games
- Interactive games
- Timer and reminders
- Animations
- Engaging Character Design
- Other: (Please specify)

d. What improvements would you like to see in the app?

e. Which part of the app does your child look forward to the most?

- Starting the app and seeing the character
- The interactive brushing activities
- Earning rewards or achievements
- Dressing up the character and leveling up
- Other: (Please specify)

f. How does your child react when earning a reward in the app?

- Does not care
- Not very excited
- Neutral
- Somewhat excited
- Very excited

g. Does your child imitate the brushing techniques shown in the app?

- Not sure

- Never
  - Rarely
  - Sometimes
  - Always
- h. How much supervision did your child need to use the app?
- None
  - Almost none
  - Neutral
  - A lot of supervision
  - Could not use the app without supervision
- i. How does your child describe the app to others (e.g., friends, family)?
- j. Is there anything your child doesn't like about the app? If so, what are they?
- k. If your child could add anything to the app, what would it be?

#### 7. Survey Questions-for parents

- a. How confident are you in your knowledge of proper brushing techniques?
- Likert scale ranging from 1 (Not confident at all) to 5 (Very confident)
- b. Before using the app, how often did you actively teach or demonstrate proper brushing techniques to your child?
- Daily
  - Several times a week
  - Weekly
  - Occasionally
  - Rarely or never
- c. Do you think the app could influence how you supervise or assist your child with brushing their teeth?
- More involvement: I'll likely have to help my child more
  - Slight increase: I may have to help a little more
  - No change: I don't foresee any change.
  - Less involvement: I'll probably have to help my child help less.
  - Independent: The app might make supervision unnecessary.
- d. Do you feel the app has provided you with new information or techniques regarding dental hygiene that you were not aware of before?
- Yes
  - No
  - Other: (Please specify)
- e. How likely are you to recommend this app to other parents?

- Likert scale ranging from 1 (Very unlikely) to 5 (Very likely)

**8. System Usability Scale (SUS)-Answers ranging from 1 (Strongly Disagree) to 5 (Strongly Agree)**

- a. I think that my child would like to use this system frequently.
- b. My child found the system unnecessarily complex.
- c. My child thought the system was easy to use.
- d. My child thinks that they would need the support of a technical person (adult) to be able to use this system.
- e. My child found the various functions in this system were well integrated.
- f. My child thought there was too much inconsistency in this system.
- g. My child would imagine that most people would learn to use this system very quickly.
- h. My child found the system very cumbersome to use.
- i. My child felt very confident using the system.
- j. My child needed to learn a lot of things before they could get going with this system.

**9. Conclusion**

- Thanking participants for their time and feedback.



## Appendix B

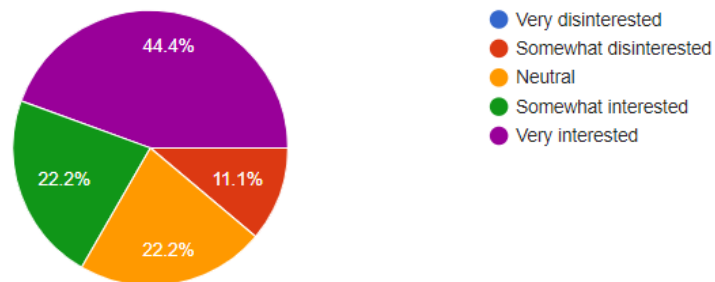
# Survey Responses

### Section 1

What was your child's initial reaction to the app?

 Copy

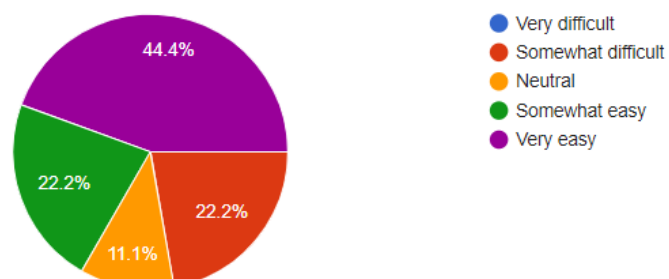
9 responses

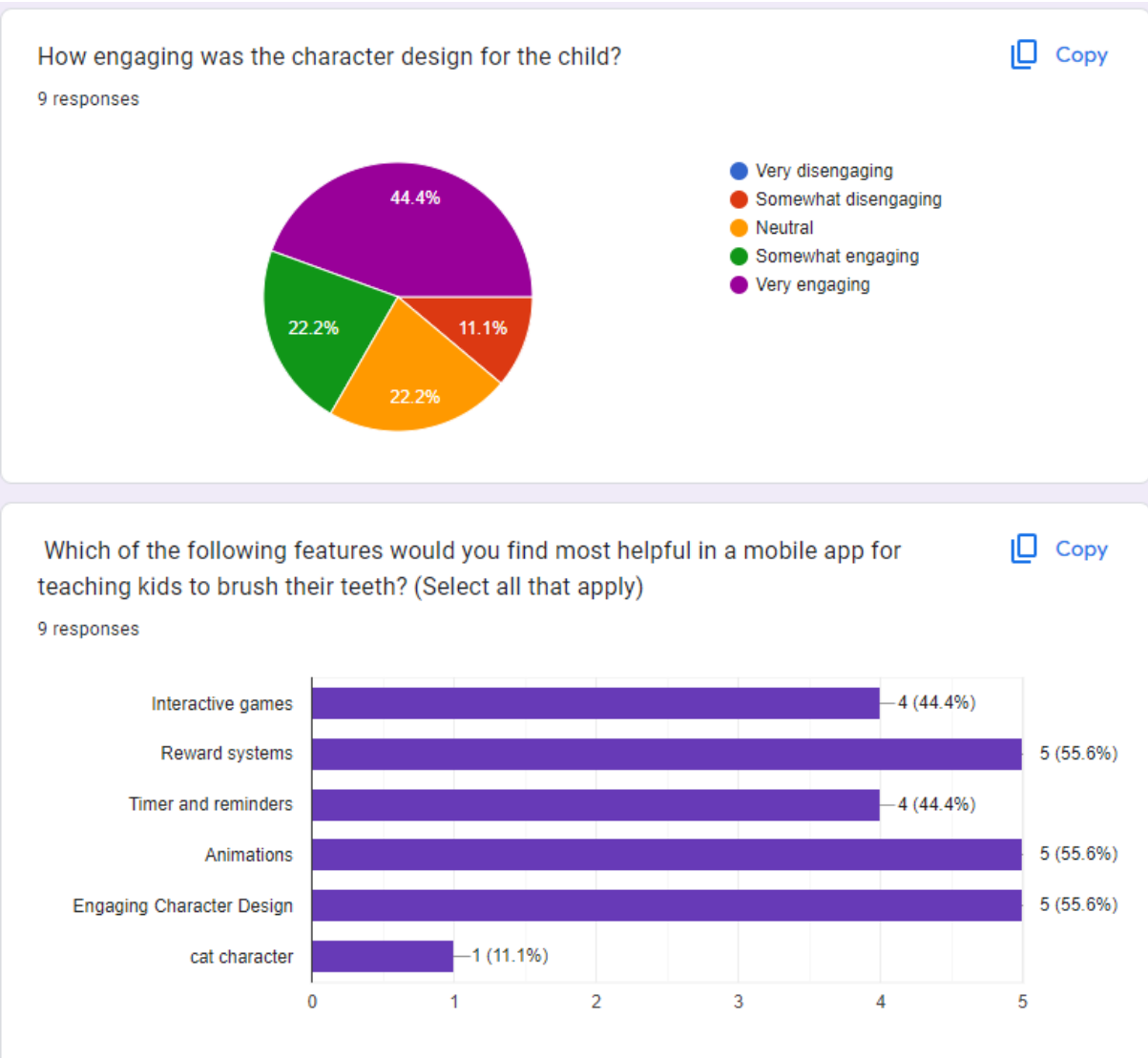


How easy was it for your child to navigate the app without assistance?

 Copy

9 responses



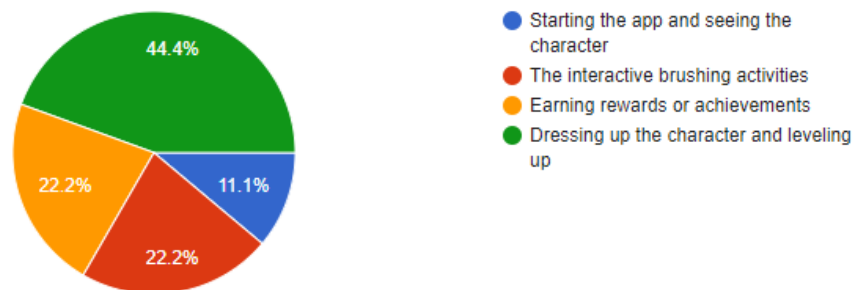


## Section 2

Which part of the app does your child look forward to the most?

 Copy

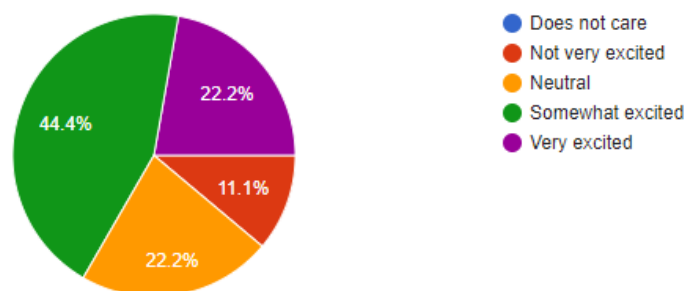
9 responses



How does your child react when earning a reward in the app?

 Copy

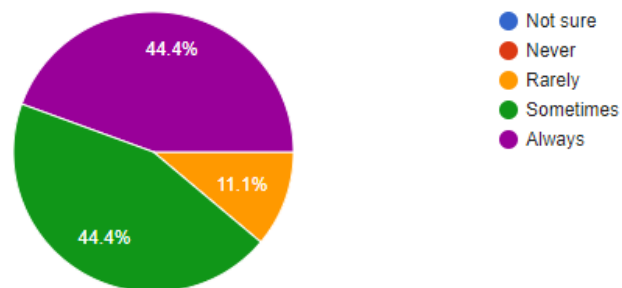
9 responses



Does your child imitate the brushing techniques shown in the app?

 Copy

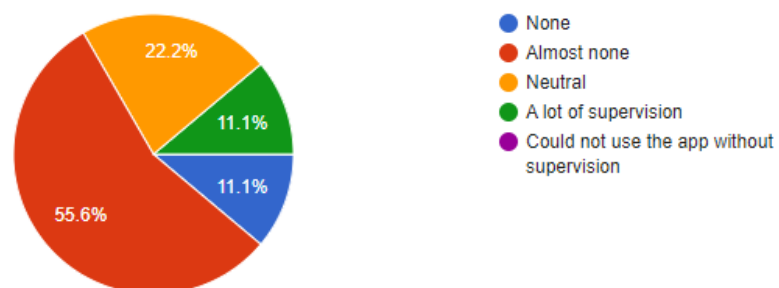
9 responses



How much supervision did your child need to use the app?

 Copy

9 responses

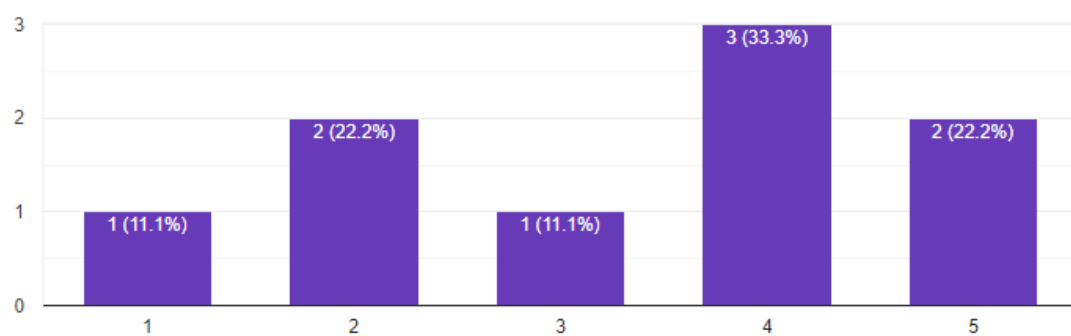


## Section 3

How confident are you in your knowledge of proper brushing techniques?

[Copy](#)

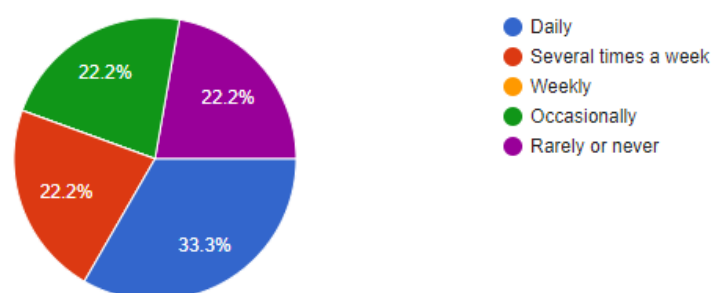
9 responses



Before using the app, how often did you actively teach or demonstrate proper brushing techniques to your child?

[Copy](#)

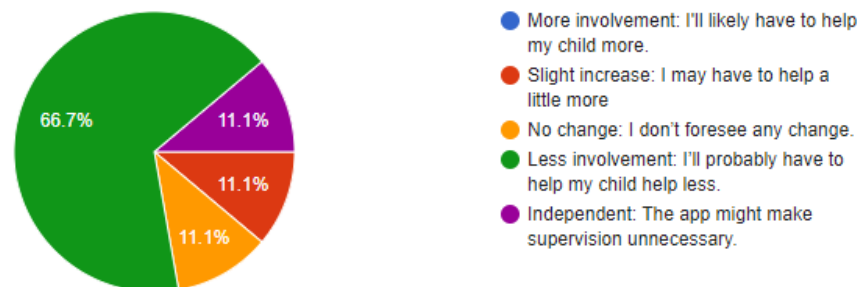
9 responses



Do you think the app could influence how you supervise or assist your child with brushing their teeth?

 Copy

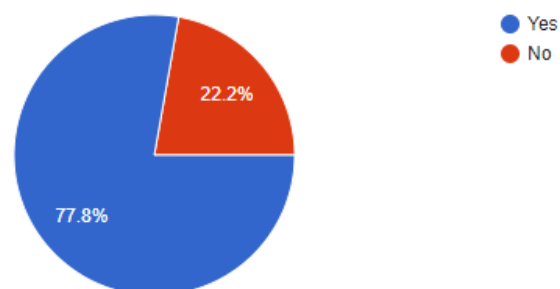
9 responses

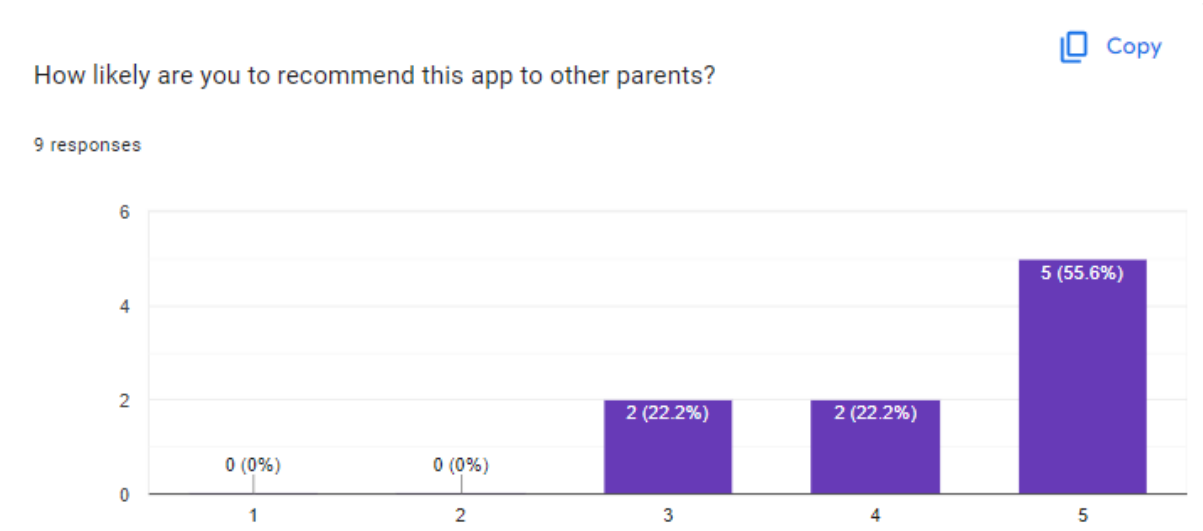


Do you feel the app has provided you with new information or techniques regarding dental hygiene that you were not aware of before?

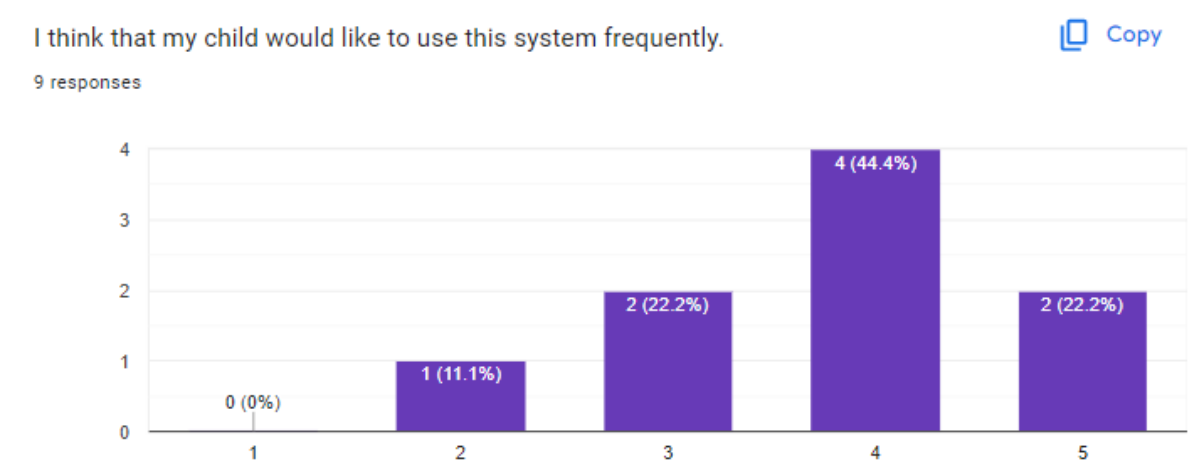
 Copy

9 responses





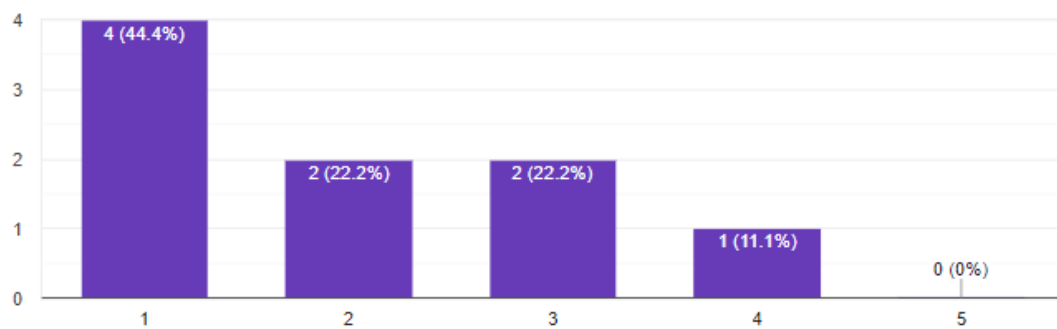
Section 4



My child found the system unnecessarily complex.

 Copy

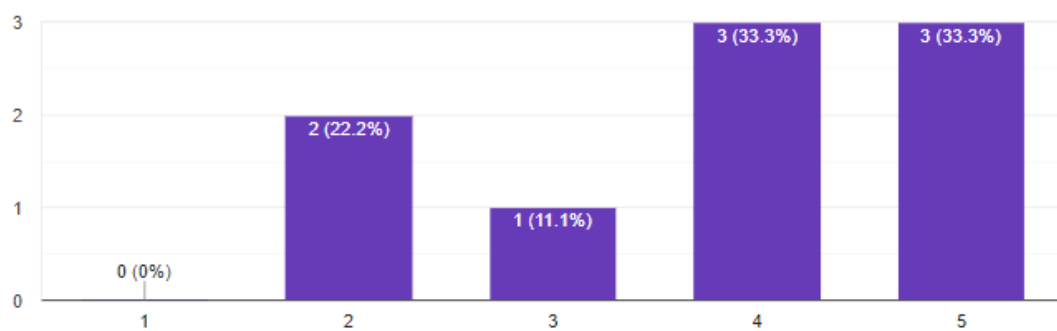
9 responses



My child thought the system was easy to use.

 Copy

9 responses

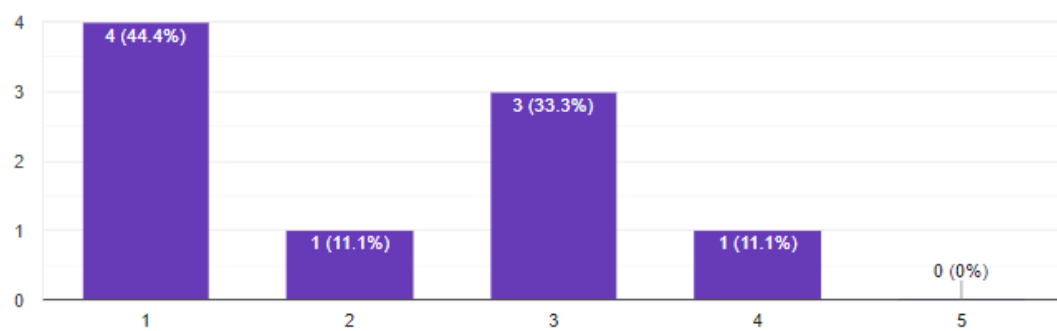




My child thinks that they would need the support of a technical person (adult) to be able to use this system.

 Copy

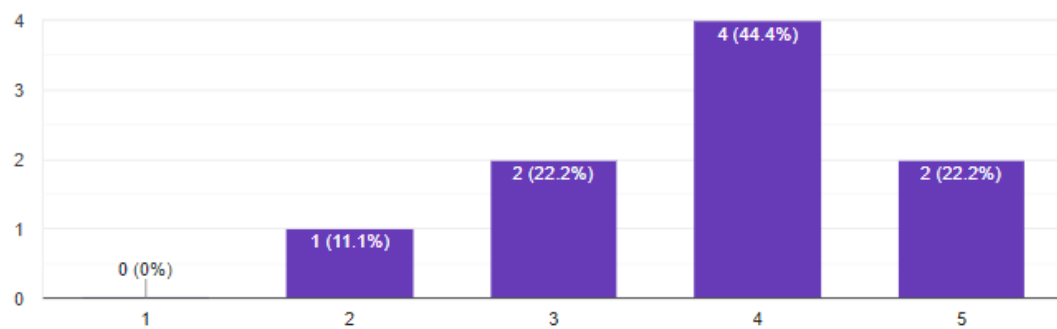
9 responses

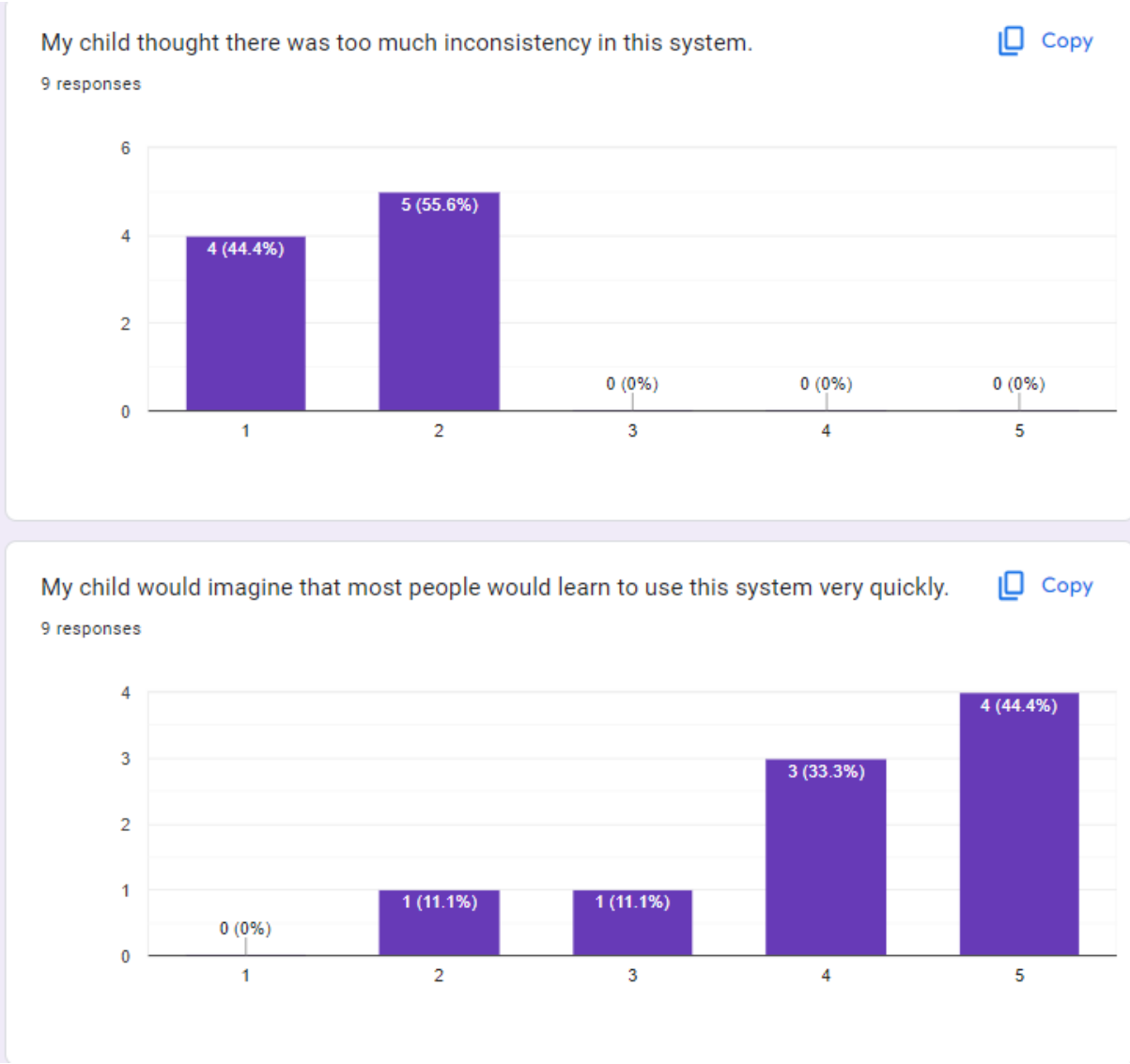


My child found the various functions in this system were well integrated.

 Copy

9 responses

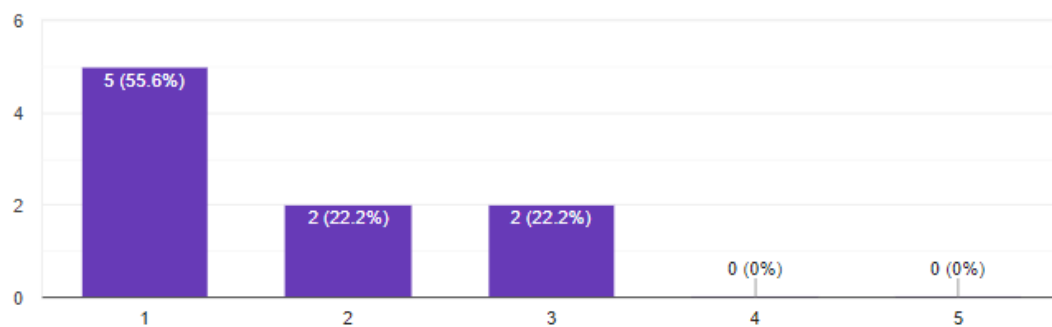




My child found the system very cumbersome to use.

 Copy

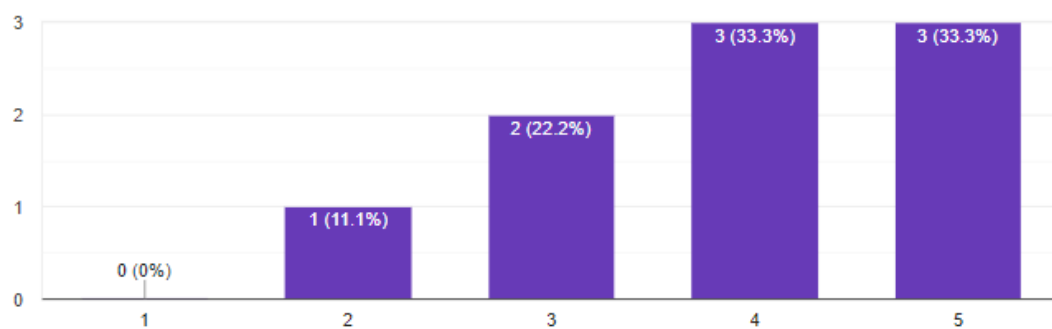
9 responses



My child felt very confident using the system.

 Copy

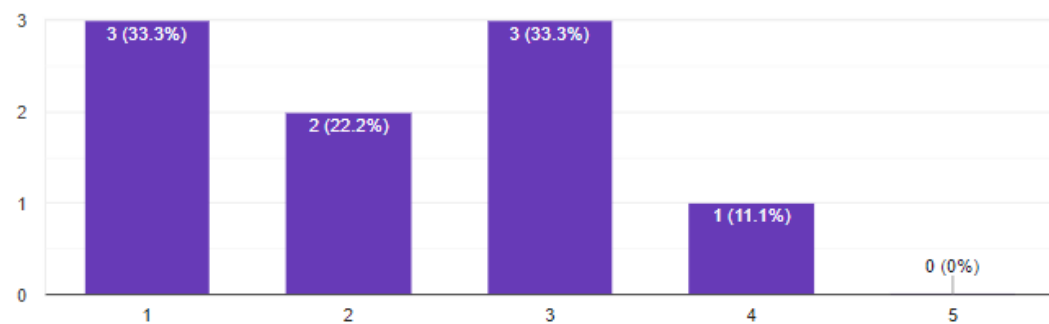
9 responses



My child needed to learn a lot of things before they could get going with this system.

 Copy

9 responses



## Appendix C

# User Manual

### C.1 Introduction

This manual provides instructions for downloading, installing, and operating Brushy.

### C.2 System Requirements

- Android Studio installed on a compatible system.
- Node.js and npm installed.
- An Android emulator setup within Android Studio.
- Visual Studio Code or any other code editor.

### C.3 Installation Instructions

#### C.3.1 Setup Android Studio

1. Download and install Android Studio from the official website.
2. Through the SDK Manager in Android Studio, install the SDK Intel x86 Emulator Accelerator (HAXM installer) version 7.6.5.
3. Create a new virtual device, recommended device: Pixel 7 Pro API 34.

#### C.3.2 Download the Application

- Open a terminal or command prompt.
- Execute the following commands:

```
git clone https://github.com/margaritaradeva/DissertationReactCLI.git  
cd Brushy
```

#### C.3.3 Install Dependencies

- In the project directory, run:

```
npm install
```

### C.3.4 Start the Application

- To start the application, run:

```
npm start
```

- Press 'a' to launch the application on the Android emulator.

## C.4 Operating Instructions

- **Sign In/Up:** Start by signing up or signing into the application using the credentials.
- **Navigating the App:** Use the bottom tab navigation to switch between Home, Progress, Shop and Parental Control screens.
- **Using Features:** Engage with various gamified features designed to teach and encourage proper dental hygiene practices.

## C.5 Troubleshooting

### C.5.1 Common Issues

- If the application fails to start, ensure the Android emulator is running correctly and that all dependencies are installed.

## Appendix D

# Maintenance Manual

### D.1 Introduction

This maintenance manual provides comprehensive information on the setup, configuration, and maintenance of the backend and frontend components of Brushy. This is intended for use by developers and maintainers to facilitate updates, bug fixes, and system enhancements.

### D.2 System Overview

The application is structured into two main components: the frontend, built with React Native, and the backend, which utilises the Django REST Framework. The backend is pre-deployed and maintained separately.

### D.3 Software Dependencies

An extensive list of all of the third-party dependencies can be found in Chapter 4 - Implementation.

#### D.3.1 Frontend

- React Native CLI
- Node.js

#### D.3.2 Backend

- Python 3
- Django
- Django REST Framework
- PostgreSQL

### D.4 Installation and Setup

#### D.4.1 Frontend Setup

- Refer to the User Manual for installation instructions of the frontend components.

#### D.4.2 Backend Setup

- Clone the backend repository:

```
git clone https://github.com/margaritaradeva/DjangoServer.git
```

- Navigate to the backend directory and install dependencies:

```
pip install -r requirements.txt
```

## D.5 File Structure

### D.5.1 Frontend

- **App.js** - Main entry point of the application.
- **src folder** - Contains all the modules, assets, and components used by the application, organised into:
  - **assets** - Stores static files like images, sounds, and fonts.
  - **core** - Contains foundational code like API connections, global state management, and secure storage.
  - **components** - Reusable React components used throughout the application.
  - **screens** - React components that serve as the main views/screens of the application.

### D.5.2 Backend

- **application folder:**
  - **models.py** - Defines the database models.
  - **serializers.py** - Handles the serialization of models for API responses.
  - **views.py** - Contains the logic for processing requests and returning responses.
  - **urls.py** - Defines the URL routes for the application.
- **core folder:**
  - **settings.py** - Configures settings for the Django project, including database configurations, security settings, and third-party apps.
  - **urls.py** - Manages the URL declarations for the entire Django project (as opposed to the application-specific urls.py).

## D.6 Making Changes

### D.6.1 Frontend Changes

- Changes to the frontend can be made locally and tested using the Android emulator.
- Commit changes to a version control system like GitHub to maintain version history:

```
git add .  
git commit -m "Describe changes here"  
git push origin main
```

### D.6.2 Backend Changes

- Backend changes are managed by the author and deployed to a live server.
- Hypothetical scenario for backend modification (if access were available):

```
# Make changes
python manage.py makemigrations
python manage.py migrate
# Commit changes to the repository
git add .
git commit -m "Describe backend changes here"
git push origin main
```

## D.7 Bug Reporting and Monitoring

- Use the provided logging and monitoring tool to track system behavior and report issues. Running this line in the terminal will provide logs for all of the requests made to the backend server and their status:

```
heroku logs --tail --app expo-brushy
```



# Bibliography

- Abijeth., B. and A.C, K. (2017). Knowledge about oral hygiene brushing techniques in children. *International Journal of Current Advanced Research*, 6:3312–3315.
- Abuhamdeh, S. (2020). Investigating the “flow” experience: Key conceptual and operational issues. *Frontiers in Psychology*, 11.
- Ahrens, D. (2015). Serious games – a new perspective on workbased learning. *Procedia - Social and Behavioral Sciences*, 204:277–281.
- Akram, S. (2021). To what extent do game design factors impact children’s engagement on digital education games?
- Alm, A., Wendt, L., Koch, G., and Birkhed, D. (2007). Oral hygiene and parent-related factors during early childhood in relation to approximal caries at 15 years of age. *Caries Research*, 42:28 – 36.
- AlMarshedi, A., Wanick, V., Wills, G. B., and Ranchhod, A. (2017). *Gamification and Behaviour*, pages 19–29. Springer International Publishing, Cham.
- Antonaci, A., Klemke, R., and Specht, M. (2019). The effects of gamification in online learning environments: A systematic literature review. *Informatics*, 6(3).
- App Store (2024). App store. Available at <https://www.apple.com/uk/app-store/>. Accessed on 23 January 2024.
- Bonnardel, N., Piolat, A., and Le Bigot, L. (2011). The impact of colour on website appeal and users’ cognitive processes. *Displays*, 32(2):69–80.
- Chang, H. Y., Park, E.-J., Yoo, H.-J., Lee, J. w., and Shin, Y. (2018). Electronic media exposure and use among toddlers. *Psychiatry Investigation*, 15(6):568–573. Accessed on 29 January 2024.
- Children’s Dental Health (CDH) Survey (2013). Child dental health survey 2013, england, wales and northern ireland. *Children’s Dental Health Survey*. Accessed: February 22, 2024.
- Deater-Deckard, K., Chang, M., and Evans, M. E. (2013). Engagement states and learning from educational games. *New directions for child and adolescent development*, 2013 139:21–30.
- Dieterle, E. (2007). Multi-user virtual environments for teaching and learning. 2:1033–1041.
- Dirks, S. J. and Monopoli, M. (2019). Oral health and healthy aging. *Healthy Aging*.
- Durin, F., Lee, R., Bade, A., On, C. K., and Hamzah, N. (2019). Impact of implementing game elements in gamifying educational environment: A study. *Journal of Physics: Conference Series*, 1358.
- Ellis, L. and Ficek, C. (2001). Color preferences according to gender and sexual orientation. *Personality and Individual Differences*, 31:1375–1379.
- Frost, R. D., Matta, V., and macivor, E. (2015). Assessing the efficacy of incorporating game dynamics in a learning management system. *J. Inf. Syst. Educ.*, 26:59–70.
- Ganss, C., Schlueter, N., Preiss, S., and Klimek, J. (2008). Tooth brushing habits in uninstructed adults—frequency, technique, duration and force. *Clinical Oral Investigations*, 13(2):203–208.

- Google Play (2024). Google play. Available at <https://play.google.com/store/search?q=brushing+teeth&c=apps&hl=en&gl=US>. Accessed on 23 January 2024.
- Green, R., Fagg, J., and Hughes, D. (2023). Children waiting over a year in pain for nhs tooth removal. Available at <https://www.bbc.co.uk/news/health-66095984>. Accessed: February 01, 2024.
- Gülcüoğlu, E., USTUN, A. B., and SEYHAN, N. (2021). Comparison of flutter and react native platforms. *Journal of Internet Applications and Management*.
- Harari, G. M., Lane, N. D., Wang, R., Crosier, B. S., Campbell, A. T., and Gosling, S. D. (2016). Using smartphones to collect behavioral data in psychological science. *Perspectives on Psychological Science*, 11(6):838–854.
- Holovaty, A. (2024). Django. <https://www.djangoproject.com/>. Accessed on 15 February 2024.
- Jesse Schell (2008). *The Art of Game Design: A Book of Lenses*. Morgan Kaufmann.
- Ketamo, H. (2015). User-generated character behaviors in educational games. pages 57–68.
- Kiatipi, M., Davidopoulou, S., Arapostathis, K., and Arhakis, A. (2021). Dental neglect in children: A comprehensive review of the literature. *The Journal of Contemporary Dental Practice*, 22(2):199–204. Accessed: January 30, 2024.
- Koster, R. (2014). *A theory of fun for game design*. O'Reilly.
- Kreuter, F., Haas, G.-C., Keusch, F., Bähr, S., and Trappmann, M. (2018). Collecting survey and smartphone sensor data with an app: Opportunities and challenges around privacy and informed consent. *Social Science Computer Review*, 38(5):533–549.
- L., D. (2023). How fast is technology growing statistics [updated 2023]. Available at <https://lefronic.com/blog/how-fast-is-technology-growing-statistics>. Accessed on 28 January 2024.
- Levine, J. (2008). Broadening our definition of gaming: Tabletop games. *Library technology reports*, 44:7.
- Lim, M., Dias, J., Aylett, R., and Paiva, A. (2009). Intelligent npcs for educational role play game. pages 107–118.
- Mackey, R., Gleason, A., and Ciulla, R. (2022). A novel method for evaluating mobile apps (app rating inventory): Development study. *JMIR mHealth and uHealth*, 10(4).
- Malone, T. (1981). Toward a theory of intrinsically motivating instruction. *Cogn. Sci.*, 5:333–369.
- NHS Inform (2024). Available at <https://www.nhsinform.scot/healthy-living/dental-health/your-teeth/teeth-cleaning-guide/>. Accessed: February 26, 2024.
- Oceana Dental (2021). Best technique for brushing teeth - cleaning your teeth. Available at <https://oceanadental.ca/blog/the-best-technique-for-brushing-teeth/>. Accessed: February 26, 2024.
- Public Health England (2022). National dental epidemiology programme (ndep) for england: Oral health survey of 5-year-old children 2022. Available at <https://www.gov.uk/government/statistics/oral-health-survey-of-5-year-old-children-2022/national-dental-epidemiology-programme-ndep-for-england-oral-health-survey-of-5-year-old-children-2022>. Accessed: January 22, 2024.
- Tezcan, A. and Richards, D. (2011). Survey of educational multi-user virtual environments and agents. pages 116–136.
- Wen, D., Chang, D. J.-W., Lin, Y.-T., Liang, C.-W., and Yang, S.-Y. (2014). Gamification design for

- increasing customer purchase intention in a mobile marketing campaign app. pages 440–448.
- Yang, J. and Shen, X. (2022). The application of color psychology in community health environment design. *Journal of Environmental and Public Health*, 2022:1–10.
- Yıldırım, M. and Mackie, I. (2019). Encouraging users to improve password security and memorability. *International Journal of Information Security*, 18(6):741–759.
- Özbek, C. D., Eser, D., Bektaş-Kayhan, K., and Ünür, M. (2017). Comparison of the tooth brushing habits of primary school age children and their parents. *Journal of Istanbul University Faculty of Dentistry*, 49(1):33. Accessed: January 26, 2024.
- Dorđević, A. (2018). Parents' knowledge about the effects of oral hygiene, proper nutrition and fluoride prophylaxis on oral health in early childhood. *Balkan Journal of Dental Medicine*, 22(3):26–31. Accessed: January 22, 2024.