MAR IVANIOS COLLEGE (AUTONOMOUS)

Mar Ivanios Vidya Nagar, Nalanchira P. O.,
Thiruvananthapuram - 695015, Kerala, India

B.Sc. Computer Science Major Project Report

EMOCARE: EMOTIONAL WELL-BEING WEB APPLICATION FOR KIDS WITH AUTISM

A report submitted in the partial fulfillment of the requirements for the award of B.Sc. Computer Science degree

SUBMITTED BY

Flickson J	2220818
Fiza Fariya	2220802
Mohammed Nihal S	2220824
Anandhu J	2220815

Under the Guidance of

Dr. Resmi V

DEPARTMENT OF COMPUTER SCIENCE
B.Sc. Computer Science
2025

MAR IVANIOS COLLEGE (AUTONOMOUS)

Mar Ivanios Vidya Nagar, Nalanchira P. O.,
Thiruvananthapuram - 695015, Kerala, India

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the project entitled *EmoCare: Emotional Well-Being Web Application for Kids with Autism*, is a bonafide record of the project work (Phase II) done by **Flickson J** (220818**), Mohammed Nihal S** (220822), **Anandhu J** (220815), and **Fiza Fariya** (220802), in partial fulfillment of the requirements for the award of the Degree of Bachelor of Science in Computer Science program by the University of Kerala.

INTERNAL GUIDE HEAD OF THE DEPARTMENT

EXTERNAL EXAMINERS

1.

2.

ACKNOWLEDGMENT

This project would not have been possible without the unending blessings, wisdom, and strength of Almighty God, for which we are truly grateful. During this project, we have been continuously inspired by His grace.

We are thankful to our Principal **Dr. Meera George**, who gave us the platform to establish ourselves to reach greater heights.

We earnestly thank **Dr. K. Oommachan**, our Director, who always encourages us to do novel things and provides all facilities.

Our sincere gratitude goes out to **Ms. Tinu C. Philip**, the Head of the Computer Science Department, for her unwavering guidance and support during this endeavor. Her support and input have been crucial in creating an atmosphere that promotes learning and development.

We wish to sincerely thank **Dr. Resmi V,** our project guide, for her invaluable support, encouragement, and guidance. Her knowledgeable direction and unwavering encouragement were essential in determining the course of our work and guaranteeing its successful conclusion.

Our profound appreciation also goes out to the faculties of Department of Computer Science, especially **Ms. Jisha Isaac**, **Dr. Anitha K L**, and **Dr. Vinodh M R**, whose knowledge, helpful criticism, and unwavering support were essential to the accomplishment of this project.

We also wish to thank our friends, classmates, and family for their unwavering understanding, encouragement, and support during this project. We persisted in finishing this work because of their faith in us.

Our deepest gratitude goes out to everyone who has helped and encouraged us along the way. Their encouragement and support were crucial to the success of this project.

ABSTRACT

EmoCare is an innovative web application designed to offer comprehensive emotional support and assistance to children with autism. Developed with contemporary web technologies, this innovative platform focuses on enhancing emotional intelligence, promoting self-regulation, and improving overall well-being. A fundamental aspect of EmoCare is its application of advanced machine learning methods to conduct real-time facial expression analysis. This allows the application to identify a broad spectrum of emotions while providing users the option to manually amend errors through intuitive visual indicators, ensuring a personalized and precise experience suited to individual requirements. The application features a diverse array of emotion management activities designed to provide both an educational and enjoyable experience. These activities encompass an Emotion Matching Activity to assist users in identifying and associating emotions, guided Breathing Exercises to facilitate relaxation and self-regulation, and calming Colouring Pages that encourage creativity while calming the mind. Additionally, a Water Penguin activity promotes healthy habits, while playlists of calming music offer an auditory escape to a peaceful state of mind. EmoCare includes a Bubble Pop Activity for engaging stress relief and a Visual Schedule feature to assist users in creating routines, providing both structure and predictability. EmoCare is carefully designed with vibrant illustrations, engaging audio, and a user-friendly interface to help children understand, express, and manage their emotions more effectively. By focusing on the distinct emotional needs of children with autism, EmoCare seeks to foster a supportive and inclusive environment that encourages emotional development and self-awareness. In summary, EmoCare is not merely an application; it is a holistic emotional support platform aimed at positively influencing the lives of its users. EmoCare aims to improve emotional well-being through its carefully selected tools and features, promoting a positive and healthy approach to emotion management, and assisting children in confidently and joyfully navigating their emotional experiences.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Full Form
CNN	Convolutional Neural Network
ML	Machine Learning
PCA	Principal Component Analysis
ASD	Autism Spectrum Disorder
GUI	Graphical User Interface
RGB	Red, Green, Blue (color model)
CSV	Comma-Separated Values
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
RAM	Random Access Memory
GHz	Gigahertz
CPU	Central Processing Unit
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
GB	Gigabyte
UI	User Interface
PHP	Hypertext Preprocessor
DOM	Document Object Model
SPA	Single Page Application

1. INTRODUCTION

An individual's capacity for social interaction communication and sensory processing is all impacted by autism spectrum disorder (ASD). Difficulties with emotional expression processing social interactions and participation in everyday activities are characteristics of the condition. Along with repetitive behaviors and focused interests, children with autism spectrum disorder (ASD) frequently exhibit increased sensory sensitivity. Each person on the autism spectrum has different strengths and challenges due to the range of severity. Having trouble controlling their emotions is a major problem for kids with autism. The capacity to identify communicate and control emotions in a way that is acceptable to others is known as emotional regulation and it is an essential skill for both social interaction and personal growth. Children with autism who have trouble controlling their emotions may experience increased anxiety heightened frustration and disruptive behavior. These symptoms can impair their general development and have an impact on their interpersonal relationships. EmoCare is a web application created to offer individualized emotional support and direction to kids with autism spectrum disorder to help them deal with these difficulties. This application aims to address the emotional development issues that children with autism spectrum disorder (ASD) face by providing a specialized platform that helps them recognize and control their emotions. EmoCare specifically manages a range of emotions, including anger, fear, joy, sadness, surprise, and neutral states, offering targeted interventions to help children better understand and regulate their feelings. Personalized feedback and support are provided by the EmoCare application which uses technologies like machine learning and real-time emotion recognition to recognize and react to user emotions. A range of activities are available through EmoCare that are intended to improve emotional intelligence and advance general health and well-being. A Activity that matches emotions, breathing techniques, colouring pages that promote relaxation and carefully selected music playlists are among the activities. Particularly for younger users the application's user-friendly interface and lively graphics significantly improve the user experience. By creating a supportive environment and enhancing emotional well-being EmoCare helps kids with autism better comprehend communicate and control their emotions. The goal of the application is to enhance the emotional health of kids with autism spectrum disorder (ASD) by providing a thorough method of emotional development that facilitates efficient emotion regulation. This project focuses on Web Application with Machine

Learning highlighting the use of cutting-edge technologies to develop a novel solution especially made to address the emotional needs of kids with autism.

1.1 OBJECTIVE

The main goal of this project is to create a user-friendly application that helps kids with autism identify, understand, and control their emotions. A child's ability to recognize and express their emotions plays a crucial role in their emotional and social development, and this application aims to support and enhance that ability through an engaging and dynamic interface. The application will provide a variety of tools and activities designed to improve emotional intelligence, enabling children to manage their emotions effectively. Considerable attention is also given to designing an aesthetically pleasing and intuitive user interface that appeals to children, ensuring the application is both accessible and enjoyable to use. By encouraging active participation and promoting a sense of comfort and ease, the user-friendly design will assist in the development of emotional awareness and regulation skills in a fun and educational way.

1.2 SCOPE OF THE PROJECT

The inability to identify and control emotions is a common problem for people with autism, and it can significantly affect their general well-being and quality of life. Their ability to fully engage in daily activities may be hampered by these emotional difficulties, which may lead to increased anxiety, frustration, and social interaction difficulties. This has led to a need for easily accessible resources to help autistic children manage their emotions. The goal of this project is to close this gap by developing an online tool that assists users in recognizing, understanding, and controlling their emotions. The app will offer several features, including emotion recognition and customized regulation methods, to help users improve their emotional intelligence and coping mechanisms in a fun and approachable way. The program's user-friendly interface, which is specially made for kids with autism, guarantees simple navigation and a pleasurable experience. Features like guided relaxation exercises, emotion identification activities, and personalized emotional feedback will be included to help users better understand their emotional states. By offering these resources, the application hopes to improve social interactions, promote self-regulation, and strengthen emotional resilience—all of which will benefit the general emotional development and well-being of kids with autism.

1.3 IDENTIFICATION OR REORGANIZATION OF NEED

For children with intellectual disabilities, particularly those diagnosed with autism, emotional regulation is a critical skill for their development and overall well-being. Social interactions and quality of life can be negatively impacted by the significant challenges that children with autism often face in identifying and managing their emotions. Despite the importance of emotional support, existing resources often fail to meet the unique needs of this group, and there is a noticeable lack of accessible and approachable resources designed to improve the emotional well-being of children with autism. This project aims to address this gap by creating a specialized application that provides emotional support and guidance. The application will prioritize a simple, intuitive interface tailored specifically for children with autism, ensuring smooth interaction and navigation. The goal is to offer a platform that helps children recognize, understand, and regulate their emotions through engaging and educational activities. By focusing on ease of use and accessibility, the application will promote self-regulation, emotional development, and overall emotional well-being for children with autism.

1.4 UNIQUE FEATURES OF THE SYSTEM

Several special features are built into the system to help kids with autism better control their emotions. Emotion identification tools which use machine learning and visual aids to identify and interpret the user's emotional state are among the main features. These features allow the application to give the child personalized support and instant feedback. Another crucial component is the use of individualized regulation strategies such as interactive activities and guided breathing exercises to assist kids in effectively and calmly controlling their emotions. These entertaining exercises provide practical methods for controlling emotions. The system also prioritizes a user-friendly design guaranteeing a straightforward and easy-to-use interface that supports children especially those with autism in navigating it. The eye-catching visual arrangement promotes engagement and improves the whole experience. Collectively these characteristics offer a comprehensive strategy for emotional growth encouraging learning and self-control in a fun and approachable manner.

2. LITERATURE REVIEW

Noteworthy applications in psychology, marketing, and human-computer interaction have led to great strides in the detection of human emotions through machine learning. These systems, which frequently use physiological cues, voice tone, and facial expression analysis, are extremely helpful in comprehending human emotions and facilitating more individualized interactions between people and technology. In the EmoCare project, machine learning is essential because it enables real-time facial expression recognition, which allows for the identification of a variety of emotions in autistic children. This ability is crucial for creating emotional support tools that work and can be tailored to each user's unique needs. By utilizing machine learning algorithms, EmoCare seeks to improve self-regulation, increase emotional intelligence, and give kids the skills they need to better comprehend and control their emotions—all of which will contribute to their general well-being. This is in line with the expanding trend of developing more responsive user-centric assistive technologies by integrating emotional recognition systems.

Real-time emotion recognition using convolutional neural networks (CNNs) was the subject of one study, which is extremely pertinent to applications such as EmoCare. After a model was trained on a variety of datasets, including 20,000 photos, the system was able to accurately classify emotions like surprise, fear, joy, and sorrow. Utilizing strategies such as data augmentation, the system enhanced performance, proving the importance of strong datasets and adjusted hyperparameters in raising the precision of real-time emotion detection. New approaches to comprehending and interpreting human emotions are provided by the suggested methodology, which highlights the possibility of wider applications in domains like virtual reality and affective computing [1].

Improved granularity in identifying emotional states has been made possible by the integration of deep learning technology into emotion recognition systems. Seven different emotions—disgust, fear, anger, sadness, surprise, and neutral—were categorized using a method that also divided them into positive, negative, and neutral scores. This model was further incorporated into an application that advances previous systems that were restricted to a smaller number of emotions by visualizing emotion recognition results through graphical representations. These developments help users receive better feedback and comprehension, which is consistent in increasing the number of useful applications of emotion analysis [2].

The fields of image processing and human-computer interaction are now actively researching facial emotion recognition. One study concentrated on creating a system that uses live webcam feeds to automatically detect emotions. Using techniques like face detection, feature extraction, and classification, it was possible to identify emotions like joy, sorrow, and rage. By assigning users particular musical tracks based on identified emotions, the study innovatively integrated music therapy and offered a fresh method of stress relief. The significance of utilizing real-time data for interactive and therapeutic applications is highlighted by this system [3].

Real-time emotion detection systems have advanced further by combining live feeds and pre-existing image datasets to analyze facial expressions. An investigation used open-source tools such as NumPy, OpenCV, and Python to create a reliable model that could predict emotions using training and testing datasets. By using various deep-learning techniques to improve accuracy, this system proved to be scalable and useful for real-world applications. These developments demonstrate the increasing capacity of emotion recognition technologies to enhance communication between humans and computers [4].

The less well-known link between water insecurity and mental health has also been investigated. Seven possible mechanisms, including material deprivation, social shame, and institutional injustices, were identified in a systematic review as connecting water insecurity to mental illness. Existing studies emphasize the necessity of empirical research to confirm causality, even though they are primarily qualitative or ethnographic. Tracking mental health indicators can provide fresh perspectives on the effectiveness of development interventions, and understanding these relationships is essential for reaching the Sustainable Development Goals about water and health [5].

The effectiveness of a Python and OpenCV-based facial emotion recognition system was shown in another study. The system's ability to accurately predict emotions by comparing testing and training datasets validated the methodology. Modern emotion detection solutions are flexible and affordable, as demonstrated by their emphasis on utilizing open-source frameworks and integrating live image analysis [6].

The development of mobile applications targeted at improving emotional intelligence is a result of efforts to address social and emotional deficiencies in children with autism spectrum disorders (ASD). In line with EmoCare's mission to provide fun interactive resources for emotional development, a systematic review identified apps that assist kids with

ASD in practicing identifying and controlling their emotions. These apps give kids innovative and approachable ways to control their emotions in authentic settings in addition to teaching them how to recognize and react to emotions. This study emphasizes how technology can promote inclusivity and emotional development [7].

Children with ASD have also been taught to recognize and communicate their emotions through the use of emotion recognition technologies. Through the use of tangible user interfaces and emotion recognition systems, one creative project allowed kids to engage with objects and facial expressions in a natural way. Positive results from the system's evaluations showed that it could be used as a teaching and therapeutic tool to help children with ASD develop their social-emotional abilities. The increasing significance of user-centric and interactive technologies in special education is reflected in this approach [8].

The improvement of reading comprehension and emotional development have been the main goals of educational applications created for students with ASD. To improve reading habits, phonological development, and vocabulary, for example, an app-based solution integrated research-proven techniques. A pilot study showed how well the app supported learning objectives and offered insightful information to educators and developers. To support a range of learning needs, such initiatives highlight how crucial it is to match pedagogical content with technological functionality [9].

Improvements in emotion recognition frameworks have made it easier for kids with ASD to diagnose them early and provide therapeutic interventions. High accuracy in identifying emotions like joy, sadness, and rage was attained by a real-time system that used deep convolutional neural networks. Fog and IoT technologies further improved the system's scalability and responsiveness, highlighting the potential of combining cutting-edge algorithms with assistive technology to help people with ASD [10].

The potential of emotion recognition systems to improve human-computer interaction is being investigated more and more. In one study, a model for identifying the six fundamental emotions of happiness, sadness, anger, fear, surprise, and disgust was trained using facial expression data. To attain high classification accuracy, the system used a hybrid model that combined deep learning architectures with feature extraction techniques. The framework has been utilized in fields like interactive educational tools and virtual assistants by incorporating real-time processing capabilities, showcasing its contribution to the advancement of technology for comprehending human emotions [11].

Emotion recognition has been further explored using multi-modal approaches that combine facial expressions with speech and physiological signals. To increase accuracy in identifying subtle emotional states, a neural network was used to integrate these modalities into a proposed system. By addressing the drawbacks of single-modal systems, this method offers a more thorough comprehension of emotional dynamics. Its potential uses in mental health monitoring and therapeutic settings demonstrate how adaptable multi-modal emotion recognition frameworks are [12].

Transfer learning approaches have been used to improve the development of real-time emotion recognition systems. By using emotion datasets to train an existing convolutional neural network model, a study was able to maintain high accuracy while drastically cutting down on training time. Transfer learning was used to expedite the development process and increase the system's usability for smaller-scale projects. These developments have created the possibility of implementing emotion detection systems in settings with limited resources, like low-budget educational institutions or rural healthcare facilities [13].

Another creative study used emotion recognition technology to improve emotional engagement in online learning environments. During virtual lessons, the system observed students' facial expressions and modified its content delivery to reflect the identified emotional states. For example, signs of frustration led to more explanations, while moments of boredom prompted more interactive activities. This framework for adaptive learning showed how emotion-aware systems can enhance learning outcomes and increase the responsiveness and efficacy of virtual classrooms [14].

Smarter settings that can recognize and react to human emotions have been made possible by the Internet of Things integration with emotion recognition systems. The use of such a system in a smart home environment was demonstrated in a case study, where emotions affected temperature, lighting, and music choices. The user and devices interacted seamlessly to create a customized living environment that enhanced comfort and general well-being. This application exemplifies the expanding movement to incorporate emotional intelligence into commonplace technology [15].

Emotion recognition has also been incorporated into driver safety systems. To identify stress or anger, a real-time system tracked drivers' physiological signals and facial expressions. This allowed for the activation of calming interventions like guided breathing

exercises or relaxing music. In addition to improving driver concentration, this preventative strategy made the roads safer. The study emphasizes how incorporating emotion recognition technologies into public safety campaigns has an impact on society [16].

From personalized environments to safety-focused interventions, the reviewed studies highlight the adaptability of emotion recognition systems in a variety of applications. These developments demonstrate how assistive technologies like EmoCare, which attempts to address the emotional and developmental difficulties experienced by kids with autism, can incorporate emotional intelligence. EmoCare expands on these developments by integrating real-time facial expression recognition, captivating activities, and user-friendly interfaces to offer flexible and user-focused solutions that promote emotional intelligence, self-control, and well-being. These revelations offer the EmoCare project a strong basis, opening the door for additional research and significant advancements in the field of emotion-centric technologies.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The majority of the physical techniques used in the current systems for providing emotional support to children with autism are caregiver interventions and conventional therapy sessions. Although there are some general-purpose apps available they are frequently not designed to meet the unique requirements of kids with autism especially when it comes to identifying emotions and offering quick entertaining activities to promote emotional control. Simple tools like emotion-based activities or schedulers that assist kids in better understanding and managing their daily routines are also lacking in many applications. This demonstrates the need for specialized products like EmoCare that target these particular areas in an approachable and user-friendly way.

3.2 PROPOSED SYSTEM

The goal of the suggested system is to fill the gaps left by current general-purpose solutions by offering customized emotional support created especially for kids with autism. Fundamentally, the system uses OpenCV-powered real-time facial expression recognition to identify a variety of emotions. With the help of this feature, a more individualized approach is made possible as the application can recommend suitable activites or activities depending on the identified emotional state. The approach makes sure that emotional support is appropriate and successful by concentrating on the special requirements of kids with autism.

The system includes a range of interactive emotion management exercises to improve its functionality even more. In addition to entertaining kids, these activities are meant to help them learn self-control in a positive setting. Examples include activities that match emotions, relaxing activities, and other fun resources that help kids understand and control their emotions. By providing a harmonious blend of entertainment and therapeutic benefits, these features hope to promote emotional development in a fun and organic manner.

The system is designed with an easy-to-use interface that is kid-friendly and puts usability first. Children can easily use the application with little help thanks to its clear graphics, calming music, and well-thought-out layout. For kids with autism, these components

are especially crucial because they create a welcoming and encouraging environment that is sensory-friendly. Through the integration of cutting-edge technology and an emphasis on user-centric design, the suggested system provides a workable and efficient way to improve the emotional health of kids with autism.

3.3 FEASIBILITY STUDY

EmoCare's feasibility study seeks to determine if the solution is a financially viable venture. It looks at the possible advantages of offering emotional support to kids with autism making sure that the system advantages outweigh the expenses of its conception creation and deployment.

3.3.1 Technical Feasibility

The integration of multiple cutting-edge technologies to produce an efficient and intuitive application is the main focus of EmoCare's technical viability. To accurately identify a variety of emotions from facial cues the system will use OpenCV for real-time facial expression recognition. This well-known and extensively utilized technology guarantees dependable performance for emotional detection in computer vision tasks. EmoCare will also have an eye-catching and captivating user interface with kid-friendly animations and vibrant graphics. Interactive activities designed to promote emotional control will improve the user experience by offering amusement and educational opportunities. Because of their intuitive design, these activities are easy for young users—especially those with autism—to navigate and interact with. Because web development technologies will be used in its development the application will be easily accessible on a variety of devices. HTML5 CSS and JavaScript are examples of contemporary web development languages and frameworks that can support the dynamic elements and interactive features required for this application. Since EmoCare can be established using all of these easily accessible technologies it is a technically feasible project that makes use of pre-existing software and development tools.

3.3.2 Social Feasibility

EmoCare's social viability is based on its capacity to meet autistic children's urgent need for emotional support. Giving them tools to help them manage their emotions is crucial because this group frequently has trouble controlling their emotions and interacting with others. EmoCare provides a solution that can significantly improve their emotional health and give children and their caregivers the much-needed support they need. EmoCare is anticipated

to be well received by therapists, educators, and caregivers due to the growing awareness of autism and the increasing need for specialized support tools. These organizations understand how critical emotional development is for kids with autism and are constantly looking for innovative ways to help them develop emotionally. Children can learn to better understand and control their emotions with EmoCare's engaging interactive features and customized activities. By providing a solution that specifically addresses the emotional needs of kids with autism and encourages their involvement in stimulating instructive and developmentally beneficial activities EmoCare also fosters inclusivity. By focusing on emotional well-being EmoCare contributes to the creation of a more inclusive environment where children with autism can feel supported valued and capable of expressing themselves effectively.

3.3.3 Operational Feasibility

The foundation of EmoCare's operational viability is its straightforward setup and intuitive user interface which make it simple to incorporate into kids' and caregivers' daily schedules. Because it requires little training the application is usable by its target audience without requiring technical expertise or specialized knowledge. Caregivers can quickly learn how to use the application to support emotional development and children can easily navigate the system. The user-friendly design of the system makes it efficient right away giving its users a smooth and pleasurable experience. The application takes into account the particular requirements of each group providing a seamless and entertaining experience for all users because it was created with both kids and caregivers in mind. This ease of use guarantees that there won't be any obstacles or issues when the application is adopted. EmoCare's scalable architecture also allows for future feature additions and updates. The system can adapt to new demands or developments in emotional support guaranteeing its continued applicability and worth. As new issues in the field of autism support arise EmoCare's scalability guarantees that it will continue to be beneficial and effective while facilitating ongoing improvement.

3.3.4 Economic Feasibility

EmoCare's economic viability is bolstered by the utilization of low-cost proprietary software and economic development techniques. For facial expression recognition and other web development technologies, the project uses open-source tools like OpenCV to reduce development costs. Because these open-source resources offer excellent functionality without requiring pricey proprietary licenses the project's overall cost stays low. Furthermore, the cost-effectiveness of the system is further enhanced by the use of reasonably priced

proprietary software. The budget for design development and deployment is kept within control by EmoCare by choosing tools and technologies that provide the required capabilities at a fair price. With this strategy, the project can concentrate its resources on developing an efficient and user-friendly solution rather than overspending on pricey hardware or software. Additionally, there may be financial opportunities from non-profits and groups that support kids with autism. Grants or other financial support may be available from these organizations for a project that meets such a pressing need for emotional support. With this outside funding, EmoCare may be able to reduce expenses and become a long-term financially viable project guaranteeing its expansion and growth.

3.4 PROBLEM DEFINITION

Autism-related emotional and social development is severely hampered by children's frequent inability to identify understand and control their emotions. Their particular needs are not met by current solutions especially when it comes to managing quick erratic facial expressions and recognizing subtle emotional cues. These flaws make it more difficult to identify emotions accurately and restrict opportunities for emotional development which exacerbates social isolation and frustration. To fill these gaps our platform uses sophisticated video capture and processing techniques analyzing multiple frames using a CNN model to reliably detect emotions. Extra features like visual schedules and customized activities improve organization and engagement while offering easily accessible specialized support to help children with autism develop their emotional intelligence and coping mechanisms.

4. SYSTEM SPECIFICATIONS

4.1 SOFTWARE REQUIREMENTS

The application will use HTML, CSS, and JavaScript on the front end to create an engaging and responsive user interface. These web technologies will guarantee that the application is both aesthetically pleasing and user-friendly. The Bulma framework will be employed to improve the design and have a clean, contemporary layout. To enable dynamic and interactive features that are crucial for user engagement, JavaScript will be utilized. The system's real-time facial expression recognition will be based on OpenCV, a robust open-source library. The application can react appropriately since OpenCV will analyze facial data to recognize various emotional states. The backend will be driven by Flask, an agile and effective web framework that guarantees seamless interaction between the front end and machine learning models. EmoCare will work on a variety of devices because it is compatible with major operating systems such as Windows, macOS, and Linux. GitHub will be used for version control, guaranteeing well-organized code management and smooth development team collaboration. This technical configuration ensures that a scalable and reliable application for kids with autism will be developed.

4.2 HARDWARE REQUIREMENTS

Certain hardware requirements must be fulfilled to guarantee EmoCare's optimal operation both during development and for end users. The purpose of these specifications is to ensure seamless functioning quick processing and a satisfying user experience.

4.2.1 Development Machine

CPU: Minimum: 2.5 GHz Processor, Recommended: 3.0 GHz Processor

Memory: Minimum: 8 GB RAM, Recommended: 16 GB RAM

Storage: Minimum: 20 GB available disk space

Graphics: Integrated or dedicated graphics card for design and development

Internet Connection: Stable connection for online resources and version control

4.3 LANGUAGE DESCRIPTION

4.3.1 HyperText Markup Language (HTML)

The fundamental language for organizing content on the web is HTML. Through a system of tags and attributes, it provides the essential building blocks required to construct a webpage, defining the format and organization of text, images, videos, and forms. By establishing the semantic structure of the content, HTML helps browsers understand the relationships between elements such as lists, headings, paragraphs, and links. The most recent version, HTML5, introduces enhanced forms for greater accessibility, the use of semantic elements, and the ability to embed multimedia content (audio and video). To further improve user experience, HTML also supports integration with JavaScript and CSS for added styling and interactivity.

HTML is one of the first things that a web developer learns, as it is the foundation of web development. It plays a crucial role in ensuring that content is presented in an organized and accessible manner. HTML is essential for SEO (Search Engine Optimization), as it allows developers to structure websites efficiently and ensures that search engines can discover content. Moreover, HTML is highly flexible and can be used to create both simple web pages and complex web applications with dynamic content. Over time, HTML has evolved to meet modern demands, such as responsive web design, which adapts content to different screen sizes on tablets and smartphones.

Developers can additionally create highly functional and interactive websites thanks to HTML's compatibility with other web technologies. Since HTML is the foundation of the internet and a critical tool for all web developers, its importance cannot be overstated.

4.3.2 Cascading Style Sheets (CSS)

The way HTML content is presented and laid out on web pages is managed by CSS, a robust stylesheet language. It is used to specify a website's appearance and feel, keeping the design and structure (which HTML provides) separate. Developers can set colors, fonts, spacing, borders, and alignment for HTML elements using CSS. More intricate styles that enhance a webpage's visual appeal, such as gradients, shadows, and transitions, are also supported by CSS. Because CSS is cascading, styles can be applied locally, globally, or individually, depending on the developer's needs.

CSS facilitates design centralization, eliminating redundancy and making it easier to maintain a website's consistent branding. Media queries for responsive design, Flexbox for complex layouts, and animations for interactive elements are just a few of the sophisticated features that CSS3, the latest version, has introduced. Thanks to these developments, developers can now create user interfaces that are captivating, responsive, and aesthetically pleasing. CSS has become essential for ensuring that websites work properly across a variety of screen sizes and resolutions, especially with the increasing use of mobile devices.

CSS preprocessors like Sass and LESS also improve the development process by offering advanced features such as variables, nested rules, and mixins. When it comes to transforming unformatted HTML into visually appealing and intuitive online experiences, CSS is indispensable. Its flexibility and power allow developers to explore a wide range of design possibilities while ensuring that web content is presented efficiently, cleanly, and visually engaging.

4.3.3 JavaScript

Developers can create interactive web elements with the help of the dynamic programming language JavaScript. JavaScript provides functionality that allows developers to build features such as forms that validate user input, interactive maps, animations, and real-time updates without requiring a page reload. This sets it apart from HTML and CSS, which mainly focus on the structure and style of content. Since JavaScript is a client-side scripting language, it operates directly within the user's browser and responds to user input instantly. This reduces delays, ensuring that websites remain responsive and engaging. Additionally, JavaScript is also used for server-side programming—especially with Node.js—which enables the development of full-stack web applications.

The event-driven programming model helps JavaScript trigger predefined actions when it detects particular events, such as mouse movements or button clicks. It plays a crucial role in developing Single Page Applications (SPAs), where it handles asynchronous operations like sending HTTP requests to retrieve data from a server without requiring a page reload. Frameworks like Angular, React, and Vue, along with libraries like jQuery, have expanded JavaScript’s capabilities, simplifying the handling of the complexities involved in modern web development. JavaScript is an essential skill for any web developer, owing to its ability to dynamically manage data and update the user interface in real-time.

JavaScript is the most popular programming language for interactive web applications due to its vast ecosystem of tools and resources, which continues to grow. This powerful and flexible language allows developers to create compelling user experiences by ensuring smooth interaction between the user and the application. As an integral part of the modern web development process, JavaScript guarantees that websites and applications are engaging, responsive, and fully interactive.

4.3.4 Bulma

Bulma is a modern responsive CSS framework whose simple minimalistic design aims to simplify web development. By utilizing Flexbox it adopts a mobile-first design ethos guaranteeing that apps are both aesthetically pleasing and useful on a range of screen sizes.

Unlike conventional CSS frameworks Bulma takes a utility-first approach which allows developers to create interfaces with predefined classes instead of requiring a lot of custom styles. Because of its modular design developers can choose just the components that are required reducing waste and improving performance. The creation of aesthetically pleasing web applications is made easier by Bulmas wide range of pre-made components which include buttons navigation bars cards and form elements. Without requiring complicated setups its Flexbox-based grid system allows for responsive and dynamic layouts.

Additionally Bulma easily integrates with JavaScript libraries and frameworks like Vue because it is a purely CSS framework. Angular Flask and js without mandated JavaScript structure. Bulma is a good choice for developers who want to create contemporary visually appealing web applications with little work because of its ease of use adaptability and lightweight design. Because of its emphasis on readability and clear syntax its a great choice for developers of all skill levels.

4.3.5 OpenCV

Designed for real-time computer vision and image processing tasks, OpenCV (Open Source Computer Vision Library) is a very strong and flexible library. It gives developers a wide range of tools and algorithms to work with visual data, allowing them to do things like image manipulation, motion tracking, object detection, and face recognition. OpenCV is essential for applications requiring visual data analysis like augmented reality,

robotics, and surveillance systems because of its real-time image and video processing capabilities.

Python, C++, and Java are just a few of the programming languages that OpenCV can be integrated with, increasing its versatility and cross-platform use. OpenCV's optimized algorithms enable it to efficiently handle large-scale image processing tasks, which is one of its noteworthy features. It has built-in tools for image analysis, filtering, and transformation, and supports a wide variety of image formats.

OpenCV also makes it simpler for developers to incorporate these features without having to start from scratch by offering pre-trained models for tasks like facial recognition. Its sizable contributor community makes sure the library is always adding new features and enhancements, maintaining its position at the forefront of computer vision technology. Because of its robustness, adaptability, and open-source nature, OpenCV is widely used in both academic research and commercial applications. As such, it is an invaluable tool for any developer working with image or video data.

4.3.6 Python

One of the most approachable programming languages for novice developers, Python is a high-level interpreted language renowned for its readability and simplicity. Python has a clear and simple syntax, which speeds up development cycles and reduces coding errors. It enables developers to use the most suitable method for a particular task by supporting a variety of programming paradigms, such as procedural, functional, and object-oriented programming. From web development and machine learning to data analysis and automation, Python's large standard library and thriving third-party package ecosystem offer developers an abundance of tools. Its widespread use in data-driven applications, artificial intelligence, and scientific computing has boosted its appeal in both the academic and business sectors.

Rapid prototyping is one of Python's greatest benefits, which makes it a great option for developing proof-of-concept applications or data-driven research projects. With libraries like TensorFlow, PyTorch, and Scikit-learn providing extensive tools for model training and data analysis, Python is also well-suited for managing challenging tasks like machine learning. Additionally, Python is frequently used in web development thanks to frameworks like Flask and Django, which make it easier to create scalable and secure web applications.

Python is a very powerful and versatile language for creating cutting-edge applications in a variety of industries because of its seamless integration with other technologies like OpenCV for computer vision. Its emphasis on simplicity and its robust ecosystem guarantee the language's ongoing applicability and expansion in the software development space.

4.3.7 Flask

Flask is a light and portable Python web framework for building web applications quickly and efficaciously. It takes a minimalist approach, which includes giving only the bare tools necessary to create a web application while giving room for developers to incorporate extra functionalities as needed.

Being a microframework, Flask lacks built-in database management and authentication systems, allowing developers to use their own tools and extensions. It relies on the Werkzeug WSGI toolkit and Jinja2 templating engine, which offer solid request handling and dynamic content generation.

Flask is a straightforward and modular architecture that is simple to implement and work with routes based on Python functions. It supports URL routing, request handling, and session management natively and is suitable for building small applications as well as large systems. Its adaptability enables smooth integration with databases such as SQLite, PostgreSQL, and MongoDB using ORM libraries such as SQLAlchemy.

Flask has extensive usage when it comes to building RESTful APIs, deployment of machine learning models, and data-driven web applications. Being minimalistic in nature but mighty in functionality makes it a favourite among developers as they can implement scalable, high-performance applications with Python.

4.4 DATASETS AND ALGORITHMS

4.4.1 Data Source

The dataset for facial expressions and emotions comes from the reputable data science and machine learning collection website, Kaggle. It is titled Autistic Children Emotions by Dr. Fatma M. Talaat. This resource is found on Kaggle and provides valuable resources for emotion recognition in children with autism.

Link to the dataset:

Autistic Children Emotions - Dr. Fatma M. Talaat -

https://www.kaggle.com/datasets/fatmamtalaat/autistic-children-emotions-dr-fatma-m-talaat

Figure 1 Sample images from the dataset (Converted to Grayscale)

4.4.2 Data Description

High-quality data is necessary for efficient model training and Dr. Fatma M. Talaat carefully selected the Autistic Childrens Emotions dataset. The RGB channels in the photos capture rich color details that improve the ability to recognize emotions. The inclusion of a wide variety of emotions enhances the models capacity to precisely recognize and categorize expressions. By varying in size the images give the training process more resilience. Eighty percent (28709 images) of the datasets 35887 images are used for training and twenty percent (167 images) are used for testing guaranteeing a fair assessment of the models performance.

4.4.3 Data Collection

The dataset is public and, hence, available for access directly downloaded from the Kaggle website. The data collection process retrieved the Autistic Children Emotions dataset, compiled by Dr. Fatma M. Talaat, which offers a diverse selection of images for various facial expressions and emotional states. This was the chosen dataset based on relevance and quality and could provide comprehensive emotion recognition within children with autism.

4.4.4 Data Preprocessing

Data preprocessing involved several steps to prepare the dataset for training. Images were resized to ensure uniform dimensions and data augmentation was applied to enhance diversity and reduce overfitting. Images were also converted to grayscale when necessary to simplify processing and focus on essential features.

4.4.5 Data Features

The dataset is categorized and includes color images with RGB values. Each image belongs to a category that reflects a particular feeling. In addition to the categorized dataset, there is an uncategorized dataset that can be used for additional training or augmentation. The large range of images the model provides enhances its ability to recognize and differentiate between various emotional expressions.

4.5 METHODOLOGY

A convolutional neural network (CNN) is a specific type of deep learning model designed for processing and categorizing image data. Its ability to automatically learn hierarchical features from images makes it especially effective for tasks like facial expression recognition. During the data collection phase, images are categorized into folders based on different emotions, and a CSV file is created to ensure that each image is accurately labeled with the corresponding emotion. In the preprocessing stage, the images are resized, normalized, and augmented to ensure consistency throughout the dataset, which enhances the model's robustness and generalization capabilities.

A CNN consists of several layers that process visual data in a structured manner. The initial layers typically focus on detecting low-level features such as edges and textures, while deeper layers combine these features to recognize more complex patterns like shapes and objects. CNNs are highly efficient because they take advantage of spatial hierarchies, meaning that the relationships between pixels are considered during the learning process. This approach allows the CNN to learn directly from raw image data, eliminating the need for manual feature extraction. During training, the network further optimizes its internal parameters to minimize prediction errors, progressively improving its accuracy in emotion prediction.

In the context of emotion recognition, deep learning models, particularly CNNs, can process large datasets of labeled images to learn complex features that distinguish different facial expressions. CNNs are especially suited for facial expression recognition tasks in real-world applications, as their adaptability enables them to handle variations in image quality, lighting, and pose. Fine-tuning a pre-trained model for emotion classification tasks by adjusting the weights to enhance accuracy is a critical step in the training process. Powerful CNN architectures allow models to capture subtle facial features, enabling them to perform tasks such as emotion detection effectively.

The model's evaluation phase involves testing its accuracy and making necessary adjustments to the parameters to improve its performance further. Deep learning, and specifically CNNs, has revolutionized computer vision by automating the feature extraction process and allowing models to learn directly from data. This method is particularly effective for emotion recognition, where subtle facial features are difficult to capture manually. CNNs have become a key tool in modern emotion recognition systems, offering a robust solution for identifying and differentiating emotional expressions with minimal human intervention.

4.6 MODEL SELECTION

A Convolutional Neural Network (CNN) model is chosen for this project and is specifically set up for emotion recognition using the dataset at hand. Because CNNs can automatically learn hierarchical features from images, they are well-suited for image classification tasks. This makes them especially useful for facial expression recognition.

The model is built using the typical CNN architecture, which consists of several convolutional layers, pooling layers, and fully connected layers. The deeper layers integrate the low-level features—such as edges, textures, and basic shapes—extracted by the convolutional layers to identify more intricate patterns and feelings. Each layer of this architecture gradually extracts more abstract information from the images, making it ideal for learning from visual data.

The ability of a CNN to adjust to the unique features of the dataset is one of its main benefits when it comes to emotion recognition. The CNN can learn to recognize subtle characteristics that differentiate between various emotions by training on a wide range of labeled images of facial expressions. As a result, the model is especially well-suited for emotion

classification tasks, such as those involving autistic children, where it is crucial to precisely recognize and interpret facial expressions.

The CNN model's efficacy for this task is further increased by its capacity to generalize across changes in lighting, pose, and image quality. All things considered, the selected CNN model is a strong and adaptable instrument for identifying emotions, guaranteeing that the system can use the data at hand to learn and adjust to the particular requirements of the project.

4.7 FEATURE ENGINEERING

During the feature engineering stage, several crucial methods are used to improve the CNN model's capacity for emotion recognition. Initially, important facial features like the mouth, nose, and eyes are recognized as facial landmarks. The CNN can concentrate on particular facial areas that are most suggestive of emotional expressions thanks to these landmarks.

To capture minute variations in facial expressions that distinguish between emotions, emotion-specific features are extracted, such as the curvature of the mouth and the arch of the eyebrows. The model is then given texture features to help it differentiate between various facial expressions. The texture and shape of facial regions are captured by these features, which is particularly crucial for identifying subtle emotional details.

CNNs excel at spotting these minute texture changes that are essential for precise emotion classification because of their capacity to recognize local patterns and spatial hierarchies. Dimensionality reduction methods are used to control the high dimensionality of image data. While keeping the most crucial information, these techniques like Principal Component Analysis (PCA) decrease the number of features.

This enhances the model's performance and generalization by increasing computational efficiency and assisting the CNN in concentrating on the most pertinent features of the data. These procedures work together to reduce the possibility of overfitting or irrelevant features while ensuring that CNN can learn to identify a variety of emotional expressions.

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The system architecture of the emotion detection platform is designed to function flawlessly through a web interface. The revised design tackles important emotion recognition issues, especially the challenge of accurately capturing a single frame that captures the emotional state of autistic children. Their facial expressions change quickly and erratically, which is the cause of this. The system now includes a video capture mechanism that reduces inaccuracies by processing multiple frames and choosing the most characteristic ones to address these issues.

Real-time video recording is supported by the platform to track and examine the user's emotional state. Only when the webcam detects a face does video recording begin, and it ends on its own after a brief period. To ensure that only high-quality frames are used for emotion recognition, the captured video frames are processed to remove any blurry or incorrectly oriented images. To get ready for analysis, these chosen frames are then put through preprocessing procedures like noise reduction, normalization, and resizing.

An emotion classification Convolutional Neural Network (CNN) model is fed the preprocessed frames. By examining several frames, the system improves the accuracy of emotion detection by identifying the predominant emotion that appears in the best-chosen frames. After detecting an emotion, the system offers a feedback loop that includes playing background music that is appropriate for the detected emotional state and presenting the user with a particular activity. More personalization is possible with manual options for choosing activities and music.

The visual schedule function on the platform, which helps users finish daily tasks, is still available. Choosing tasks is made simple by their visual representation. When a task is finished, it is crossed off the schedule, which helps users stay organized and focused. This is especially helpful for kids with autism.

The CNN model, which is implemented with the OpenCV framework and Python programming language, is at the heart of the emotion recognition process. Selected frames are improved for consistency for every video by normalizing and resizing them.

Convolutional layers, which are intended to identify low-level features like edges, textures, and fundamental facial shapes, are the first processing layers in the CNN. The most basic visual features of the input frames are extracted by these layers.

To reduce dimensionality while maintaining important information, the feature maps are downsampled by subsequent pooling layers. The model becomes more computationally efficient and dependable as a result. Following pooling and convolution, the data is compressed into a one-dimensional vector and sent through layers that are fully connected for additional processing. Anger, fear, surprise, happiness, sadness, and neutrality are among the states that the model produces a probability distribution over by classifying emotions using the learned features.

A vast collection of facial image data, helps the CNN model can efficiently learn and generalize emotional patterns. Because of its modular design, it can be continuously improved and adjusted to accommodate new data, guaranteeing consistent emotion recognition accuracy. With the help of video capture, this improved system architecture gets around the drawbacks of single-frame analysis and guarantees more accurate and dependable emotion recognition while preserving user-focused features that facilitate organization and personalization.

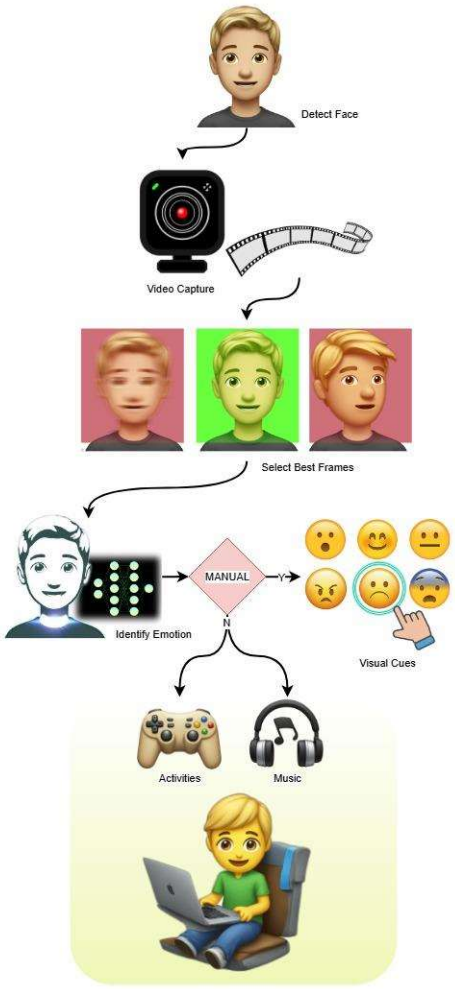

Figure 2 System Architecture

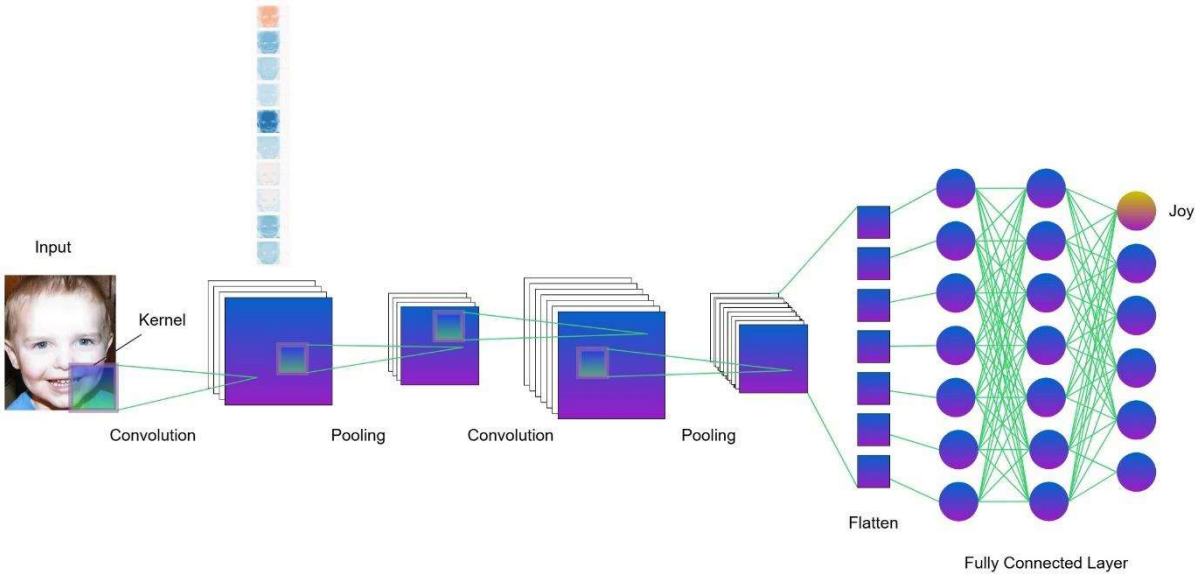

Figure 3 CNN Architecture

5.2 USE CASE DIAGRAM

The use case diagram highlights the system's primary functionalities by focusing on how a single user interacts with it. To emphasize the smooth user experience, it highlights tasks like emotion recognition activity selection and visual schedule management.

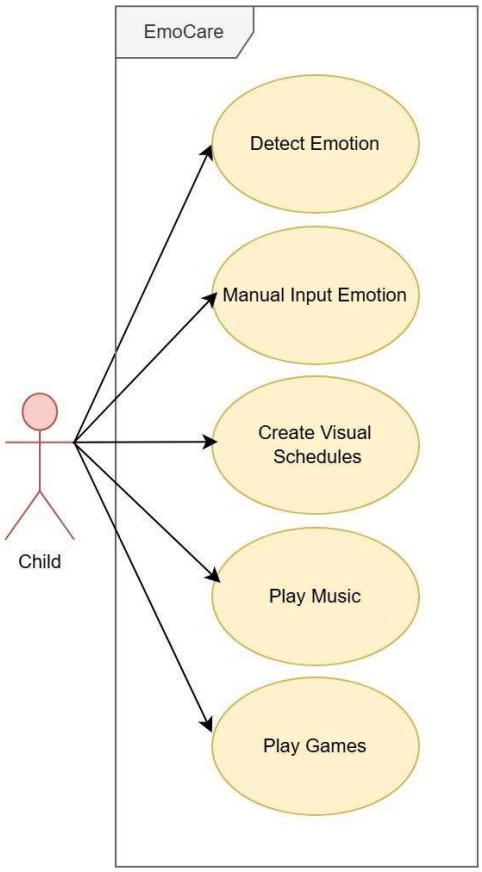

Figure 4 Use Case Diagram

5.3 MODULE DESIGN

The system's functional modules and their interactions are visually broken down in the module design diagram which is shown as a flowchart. It ensures a clear understanding of the system's operational structure by outlining the sequential flow of processes from capturing user input to emotion detection activity recommendation and task management.

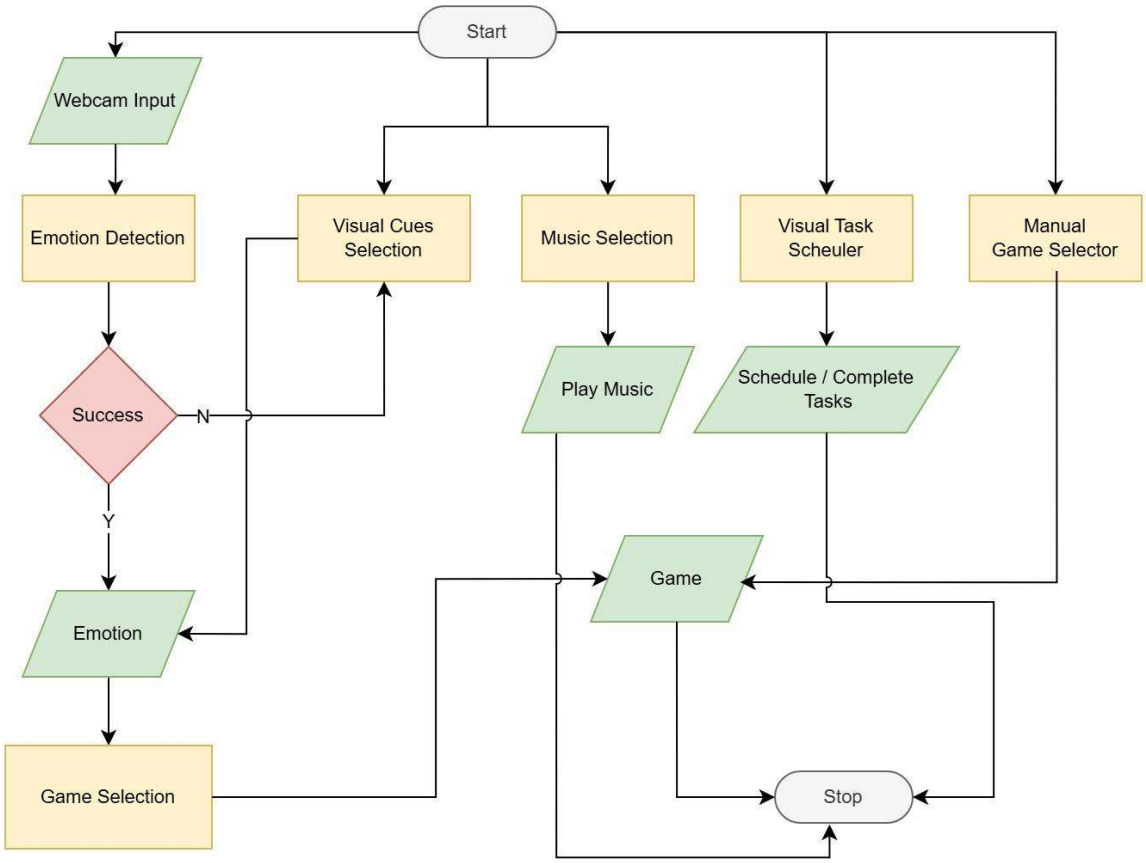

Figure 5 Flowchart representing the system

6. MODULE DESCRIPTION

The system's interconnected modules play a distinct part in improving user interaction and overall functionality. The main modules are Visual Scheduler, Emotion Detector, Activity Recommender, Music Recommender and Video Capture. When combined these modules offer a dynamic and user-focused experience by facilitating seamless and effective emotion detection task management and personalized recommendations. Each module has a thorough description below.

6.1 VIDEO CAPTURE

The Video Capture module is a critical part of EmoCare that provides real-time facial emotion recognition using an automated recording mechanism. When the application is activated, the system triggers the webcam instantly without prompting the user for any action. It keeps scanning the video stream constantly, being in wait mode until it detects a human face. After a face is detected, the module will automatically record a 10-second video with several frames to enhance the emotion analysis reliability. This method ensures that emotion recognition is done by a larger group of expressions instead of a single frame to increase accuracy. If the face is not detected, the system keeps observing until the face is detected, avoiding unwanted recordings. The hands-free nature of the application provides it with increased usability, and the kids can work with the system as they do with their fingers. This ease of video recording creates a more intuitive user interface that allows users to access emotion recognition easily, thus enhancing overall performance.

6.2 EMOTION DETECTOR

The task of the Emotion Detector module is to examine facial expressions and determine the user's predominant emotion in real-time. After the Video Capture module captures a 10-second video, the system processes the frames and performs facial recognition methodologies to determine happiness, sadness, anger, fear, surprise, and neutrality. By considering several frames, the module is able to identify the most representative emotional state instead of a one-time decision. The emotion is utilized to tailor the user experience by providing them with suitable activities or background music. This is done in an automated and transparent manner, allowing children to be provided with real-time emotional assistance without having to manually input anything, making the system intuitive and user-friendly.

6.3 ACTIVITY RECOMMENDER

he Activity Recommender module makes the experience personalized by recommending an activity based on the sensed emotion. After the Emotion Detector module detects the emotional state of the user, this module chooses a suitable activity that is intended to either reinforce positive emotions or neutralize difficult ones. If the system recognizes anger, it suggests the Bubble Pop Activity to offer a fun method to let off steam. For sadness, the Breathing and Clap Exercise assists in regulating emotions and relaxing. For fear detected, a Coloring Page is recommended to give a calming and creative solution. The Emotion Matching Activity which reinforces emotion recognition is initiated in a neutral emotional state. The Water Penguin Activity promotes constant positive interaction which is recommended for happiness. Lastly to keep the experience engaging and fun the Sound Surprise Activity is activated for surprise. By providing users with tailored activities that are appropriate for their present emotional state this mechanism fosters emotional well-being through engaging and joyful interactions.

Emotion	Activity
Angry	Bubble Pop
Sad	Breathing and Clap Exercise
Fear	Coloring Page
Neutral	Emotion Matching
Happy	Water Penguin
Surprise	Sound Surprise

Table 1 Various Emotions and Their Activities

6.4 MUSIC RECOMMENDER

The Music Recommender module adds to the user experience by providing music recommendations based on emotional states. Users can choose an emotion manually from a pre-listed set, upon which the system fetches a list of music tracks that are related to the chosen emotion. The tracks are handpicked to either calm, energize, or match the user's prevailing mood. By enabling users to customize their musical experience, the module offers

an adaptable and engaging means of emotion regulation, eliciting relaxation, concentration, or happiness depending on personal choice.

6.5 VISUAL SCHEDULER

The Visual Schedule module is a basic, picture-based to-do list that enables users to organize their activities in an easy-to-use manner. Users can choose from a pre-set list on the left, which are then added to the to-do list on the right side. When each task is accomplished, checking it off removes it from the list, giving a tangible and interactive measure of progress. This structured but adaptable strategy keeps children with autism involved, supporting routine development and task completion without the intricacy of time-based scheduling.

7. SYSTEM IMPLEMENTATION

7.1 LIBRARIES AND PACKAGES USED

This project employs a range of Python packages for deep learning, image processing, and web development. Here is a description of their functionality:

- NumPy: Facilitates numerical operations and array computations.
- Matplotlib: Offers plotting functionalities for displaying model accuracy and loss.
- OpenCV: Offers image processing and computer vision capabilities.
- TensorFlow Keras:
    - Sequential: Builds the deep learning model layer by layer.
    - Dense, Dropout, Flatten, Conv2D, MaxPooling2D: Different layers employed to create the convolutional neural network.
    - Adam: Optimizer used for model weight tuning in an efficient manner.
    - ImageDataGenerator: Augments and preprocesses image data sets to be used for training and validation.
    - load_model: Loads a pre-existing model for prediction.
    - img_to_array: Transforms images to array format for processing.
- Flask:
    - Flask: Develops a web application for serving the model.
    - Response, render_template, jsonify, request: Handles HTTP responses, templates, JSON data, and user requests.
- Collections:
    - Counter: Counts occurrences of elements in a list or array.
- OS Module: Sets system-level environment variables.
- Time: Tracks execution time for performance measurement.

Extra Libraries for Front-end and UI Improvements

- AOS.js: Allows for smooth scroll-based animations.
- Attention.css: Offers stunning animations for UI components.
- Bulma.css: A lightweight CSS framework for responsive design.
- JS-Confetti: Introduces interactive confetti animations to add celebratory effects.

7.2 CODE FOR SYSTEM DEVELOPMENT

7.2.1 Python Code for Model Creation

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
def plot_model_history(model_history):
    fig, axs = plt.subplots(1,2,figsize=(15,5))

axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accurac
y'])

axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val
_accuracy'])
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')

axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.hi
story['accuracy'])/10)
    axs[0].legend(['train', 'val'], loc='best')
    axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])

axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'
])
    axs[1].set_title('Model Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')
```

```
axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history
['loss'])/10)
    axs[1].legend(['train', 'val'], loc='best')
    fig.savefig('plot.png')
    plt.show()
train_dir = 'data1/train'
val_dir = 'data1/test'
num_train = 28709
num_val = 7178
batch_size = 64
num_epoch = 50
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(48,48),
        batch_size=batch_size,
        color_mode="grayscale",
        class_mode='categorical')
validation_generator = val_datagen.flow_from_directory(
        val_dir,
        target_size=(48,48),
        batch_size=batch_size,
        color_mode="grayscale",
        class_mode='categorical')
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer=Adam(learning_rate=0.0001,
decay=1e-6),metrics=['accuracy'])
model_info = model.fit(
        train_generator,
        steps_per_epoch=num_train // batch_size,
        epochs=num_epoch,
        validation_data=validation_generator,
        validation_steps=num_val // batch_size)
plot_model_history(model_info)
model.save_weights('model1.h5')
```

7.2.2 Python Code for Core Server Implementation

```
from flask import Flask, Response, render_template, jsonify, request
import cv2
import numpy as np
from collections import Counter
from keras.models import load_model
from keras.preprocessing.image import img_to_array
import os
from time import time
basepath = os.path.dirname(__file__)
face_classifier = cv2.CascadeClassifier(os.path.join(basepath,
"haarcascade_frontalface_default.xml"))
classifier = load_model(os.path.join(basepath, 'model.h5'))
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']
app = Flask(__name__)
dominant_emotion = None
detection_in_progress = False
```

```
detection_completed = False
def gen_frames():
    global dominant_emotion, detection_in_progress, detection_completed
    cap = cv2.VideoCapture(0)
    start_time = None
    emotions_detected = []
    while True:
        success, frame = cap.read()
        if not success:
            break
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_classifier.detectMultiScale(gray)
        if len(faces) > 0 and not detection_in_progress and not detection_completed:
            detection_in_progress = True
            start_time = time()
            emotions_detected.clear()
        if detection_in_progress:
            for (x, y, w, h) in faces:
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 255), 2)
                roi_gray = gray[y:y + h, x:x + w]
                roi_gray = cv2.resize(roi_gray, (48, 48), interpolation=cv2.INTER_AREA)
                if np.sum([roi_gray]) != 0:
                    roi = roi_gray.astype('float') / 255.0
                    roi = img_to_array(roi)
                    roi = np.expand_dims(roi, axis=0)
                    prediction = classifier.predict(roi)[0]
                    label = emotion_labels[prediction.argmax()]
                    emotions_detected.append(label)
                    cv2.putText(frame, label, (x, y), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255,
0), 2)
        if detection_in_progress and start_time and time() - start_time > 10:
            detection_in_progress = False
            detection_completed = True
            if emotions_detected:
```

```
            dominant_emotion = Counter(emotions_detected).most_common(1)[0][0]
        else:
            dominant_emotion = "none"
        print(f"Dominant Emotion: {dominant_emotion}")
    _, buffer = cv2.imencode('.jpg', frame)
    frame = buffer.tobytes()
    yield (b'--frame\r\n'
           b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')
@app.route('/get_emotion')
def get_emotion():
    global dominant_emotion, detection_completed
    if request.args.get('reset') == 'true':
        global detection_in_progress
        detection_in_progress = False
        detection_completed = False
        dominant_emotion = None
        print("Detection state reset.")
    if not detection_completed:
        return jsonify({"status": "waiting"})
    elif dominant_emotion == "none":
        return jsonify({"status": "failed"})
    else:
        return jsonify({"status": "success", "emotion": dominant_emotion})
@app.route('/happy')
def happy_page():
    return render_template('activity/happy.html')
@app.route('/sad')
def sad_page():
    return render_template('activity/sad.html')
@app.route('/angry')
```

```
def angry_page():
    return render_template('activity/angry.html')
@app.route('/surprised')
def surprised_page():
    return render_template('activity/surprised.html')
@app.route('/neutral')
def neutral_page():
    return render_template('activity/neutral.html')
@app.route('/fear')
def fear_page():
    return render_template('activity/fear.html')
@app.route('/activity')
def activity_page():
    return render_template('activity.html')
@app.route('/music')
def music_page():
    return render_template('music.html')
@app.route('/task')
def task_page():
    return render_template('task.html')
@app.route('/music_happy')
def music_happy_page():
    return render_template('music/happy.html')
@app.route('/music_sad')
def music_sad_page():
    return render_template('music/sad.html')
@app.route('/music_angry')
def music_angry_page():
    return render_template('music/angry.html')
@app.route('/music_surprised')
def music_surprised_page():
    return render_template('music/surprised.html')
@app.route('/music_neutral')
def music_neutral_page():
```

```
    return render_template('music/neutral.html')
@app.route('/music_fear')
def music_fear_page():
    return render_template('music/fear.html')
@app.route('/')
def index():
    return render_template('index.html')
if __name__ == '__main__':
    app.run(debug=True)
```

7.2.3 HTML Base Template Structure

```
<!DOCTYPE html>
<html>
    <head>
        <title>EmoCare</title>
        <link rel="stylesheet" href="{{url_for('static',filename='css/bulma.css')}}"
type="text/css">
        <link rel="stylesheet" href="{{url_for('static',filename='css/attention.css')}}"
type="text/css">
        {% block style_content %} {% endblock %}
        <meta charset="UTF-8">
                                        <meta name="viewport" content="width=device-width,
initial-scale=1">
                                        <meta name="author" content="Flickson J, Fiza Fariya,
Anandhu J, Mohammed Nihal">
                                        <link rel="shortcut icon"
href="{{url_for('static',filename='images/favicon.ico')}}" type="image/x-icon">
    </head>
    <body>
        <nav class="navbar">
                                         <div class="navbar-brand is-flex is-align-items-center"
style="width: 100%;">
                                          <a class="navbar-item u--pulse"
href="{{url_for('index')}}">
```

<img src="{{url_for('static', filename='images/logo.png')}}" alt="EmoCare Logo" style="max-height:80px" class="py-2 px-2">

{% block nav_content %}
{% endblock %}

Activities
Music
Tasks

Activities
Music

```
                    <a class="button is-primary" href="{{
url_for('task_page') }}">Tasks</a>
                  </div>
                </div>
              </div>
            </div>
          </nav>
          {% block body_content %}
          {% endblock %}
          <script>
            function toggleNavbar() {
              var nav = document.getElementById("navMenu");
              var burger = document.querySelector(".navbar-
burger");
              nav.classList.toggle("is-active");
              burger.classList.toggle("is-active");
            }
            const emotionImages = [
              {
                emotion: "Happy",
                imageUrl: "{{ url_for('static',
filename='images/Happy.gif') }}",
                activityUrl: "{{ url_for('happy_page') }}"
              },
              {
                emotion: "Sad",
                imageUrl: "{{ url_for('static',
filename='images/Sad.gif') }}",
                activityUrl: "{{ url_for('sad_page') }}"
              },
              {
                emotion: "Angry",
                imageUrl: "{{ url_for('static',
filename='images/Angry.gif') }}",
```

```
                activityUrl: "{{ url_for('angry_page') }}"
            },
            {
                emotion: "Surprised",
                imageUrl: "{{ url_for('static',
filename='images/Surprised.gif') }}",
                activityUrl: "{{ url_for('surprised_page') }}"
            },
            {
                emotion: "Neutral",
                imageUrl: "{{ url_for('static',
filename='images/Neutral.gif') }}",
                activityUrl: "{{ url_for('neutral_page') }}"
            },
            {
                emotion: "Fear",
                imageUrl: "{{ url_for('static',
filename='images/Fear.gif') }}",
                activityUrl: "{{ url_for('fear_page') }}"
            }
        ];
        </script>
        {% block script_content %}
        {% endblock %}
  </body>
</html>
```

7.2.4 HTML Home Page

```
{% extends "base.html" %}
{% block style_content %}
<link rel="stylesheet" href="{{url_for('static',filename='css/index_1.css')}}" type="text/css">
{% endblock %}
{% block nav_content %}
<div class="navbar-item is-flex-grow-1 is-flex is-justify-content-center">
```

```
    <div class="has-text-primary has-text-centered u--pulse">
      <p class="is-size-4 has-text-weight-semibold">👋 Hello there!</p>
      <p class="is-size-6 has-text-weight-medium">Please look at the camera</p>
    </div>
</div>
{% endblock %}
{% block body_content %}
<div class="loading-screen">
    <img src="{{url_for('static', filename='images/Logo.png')}}" alt="EmoCare Logo"
class="u--pulse logo">
    <div class="lds-ellipsis">
        <div></div><div></div><div></div><div></div>
    </div>
</div>
<div class="video-container">
                        <div class="lds-ellipsis">
        <div></div><div></div><div></div><div></div>
    </div>
  <img class="video-cam u--pulse" id="video-stream" src="">
</div>
<audio id="bgAudio" autoplay>
  <source src="{{url_for('static', filename='audio/welcome.mp3')}}" type="audio/mp3">
</audio>
<div class="modal" id="emotionModal">
    <div class="modal-background"></div>
    <div class="modal-content">
        <div class="box">
            <div class="has-text-centered">
                <h2 class="title is-5 u--pulse">Detected Emotion</h2>
                <img id="detectedEmotionGif" src="" alt="Detected Emotion">
                <p id="detectedEmotionText" class="is-size-4 has-text-weight-semibold mt-
2"></p>
```

```
            <p id="countdown" class="is-size-5 has-text-weight-semibold mt-2">Loading
Activity in <span id="countdown-number">10</span> seconds...</p>
        </div>
        <div class="has-text-centered mt-4">
            <p id="chooseEmotionText" class="is-size-6 has-text-weight-semibold"></p>
            <div id="otherEmotionsGrid">
            </div>
        </div>
        <div class="has-text-centered mt-4">
            <button id="reloadButton" class="button is-primary u--pulse">Detect
Again</button>
        </div>
      </div>
   </div>
</div>
{% endblock %}
{% block script_content %}
<script>
                    function adjustVideoPosition() {
                    var navbar = document.querySelector(".navbar");
                    if (navbar) {
                      var navbarHeight = navbar.offsetHeight;
                      document.documentElement.style.setProperty("--navbar-height",
navbarHeight + "px");
                     }
                    }
                    window.addEventListener("load", adjustVideoPosition);
                    window.addEventListener("resize", adjustVideoPosition);
                    document.addEventListener("DOMContentLoaded", function () {
                        var audio = document.getElementById("bgAudio");
                        var playPromise = audio.play();
                        if (playPromise !== undefined) {
                            playPromise.catch(() => {
                                document.addEventListener("click", function () {
```

```
                    audio.play();
                }, { once: true });
            });
        }
    });
    window.addEventListener("load", function(){
                setTimeout(function(){
                    document.querySelector(".loading-
screen").style.display="none";
                    document.querySelector(".video-
container").style.display="flex";

    document.querySelector(".navbar").style.display="block";
                    var videoImg =
document.getElementById("video-stream");
                    var spinner =
document.querySelector(".video-container .lds-ellipsis")
                    spinner.style.display = "block";
                    videoImg.src = "/video_feed";
                    videoImg.onload=function(){
                        spinner.style.display =
"none";
                    }
                    window.dispatchEvent(new
Event('resize'));
                },4000);
    });
    document.addEventListener('DOMContentLoaded', function () {
  const emotionModal = document.getElementById('emotionModal');
  const detectedEmotionText = document.getElementById('detectedEmotionText');
  const detectedEmotionGif = document.getElementById('detectedEmotionGif');
  const chooseEmotionText = document.getElementById('chooseEmotionText');
  const otherEmotionsGrid = document.getElementById('otherEmotionsGrid');
  const modalBackground = document.querySelector('.modal-background');
```

```
    const reloadButton = document.getElementById('reloadButton');
    let countdownInterval;
    function playEmotionAudio(emotion) {
        if (emotion.toLowerCase() === 'neutral') return;
        const audio = new Audio(`{{ url_for('static', filename='audio/') }}EmoCare ${emotion}
Flickson.mp3`);
        audio.play().catch(error => {
            console.log('Audio playback failed:', error);
        });
    }
    function fetchDominantEmotion() {
        const reset = !window.hasDetectionStarted;
        fetch(`/get_emotion?reset=${reset}`)
            .then(response => response.json())
            .then(data => {
                if (data.status === 'success') {
                    const detectedEmotion = data.emotion;
                    const detectedEmotionData = emotionImages.find(item => item.emotion ===
detectedEmotion);
                    if (detectedEmotionData) {
                        detectedEmotionGif.src = detectedEmotionData.imageUrl;
                        detectedEmotionText.textContent = detectedEmotion;
                        playEmotionAudio(detectedEmotion);
                        startCountdown(detectedEmotionData.activityUrl);
                    } else {
                        detectedEmotionGif.src = "";
                        detectedEmotionText.textContent = "No GIF found for this emotion.";
                    }
                    chooseEmotionText.textContent = `Not ${detectedEmotion}? Choose your
emotion`;
                    otherEmotionsGrid.innerHTML = "";
                    emotionImages
                        .filter(item => item.emotion !== detectedEmotion)
                        .forEach(item => {
```

```
                    const emotionDiv = document.createElement('div');
                    emotionDiv.className = "column is-one-third";
                    emotionDiv.innerHTML = `
                        <div class="has-text-centered emotion-choice" data-
emotion="${item.emotion}" data-activity-url="${item.activityUrl}">
                            <img src="${item.imageUrl}" alt="${item.emotion}" style="max-
width: 40px; height: auto; cursor: pointer;">
                            <p class="is-size-6 has-text-weight-semibold mt-
1">${item.emotion}</p>
                        </div>
                    `;
                    otherEmotionsGrid.appendChild(emotionDiv);
                });
                document.querySelectorAll('.emotion-choice').forEach(choice => {
                    choice.addEventListener('click', () => {
                        const activityUrl = choice.getAttribute('data-activity-url');
                        window.location.href = activityUrl;
                    });
                });
                emotionModal.classList.add('is-active');
            } else if (data.status === 'waiting') {
                setTimeout(fetchDominantEmotion, 1000);
            } else if (data.status === 'failed') {
                detectedEmotionGif.src = "";
                detectedEmotionText.textContent = "No emotion detected";
                chooseEmotionText.textContent = "Choose your emotion";
                otherEmotionsGrid.innerHTML = "";
                emotionImages.forEach(item => {
                    const emotionDiv = document.createElement('div');
                    emotionDiv.className = "column is-one-third";
                    emotionDiv.innerHTML = `
                        <div class="has-text-centered emotion-choice" data-
emotion="${item.emotion}" data-activity-url="${item.activityUrl}">
```

```
                    <img src="${item.imageUrl}" alt="${item.emotion}" style="max-width:
40px; height: auto; cursor: pointer;">
                    <p class="is-size-6 has-text-weight-semibold mt-
1">${item.emotion}</p>
                </div>
            `;
            otherEmotionsGrid.appendChild(emotionDiv);
        });
        document.querySelectorAll('.emotion-choice').forEach(choice => {
            choice.addEventListener('click', () => {
                const activityUrl = choice.getAttribute('data-activity-url');
                window.location.href = activityUrl;
            });
        });
        emotionModal.classList.add('is-active');
    }
})
.catch(error => {
    console.error('Error fetching emotion:', error);
});
window.hasDetectionStarted = true;
}
function startCountdown(activityUrl) {
let countdown = 10;
const countdownElement = document.getElementById('countdown-number');
if (countdownInterval) {
    clearInterval(countdownInterval);
}
countdownInterval = setInterval(() => {
    countdown--;
    countdownElement.textContent = countdown;
    if (countdown <= 0) {
        clearInterval(countdownInterval);
        window.location.href = activityUrl;
```

```
            }
        }, 1000);
    }
    reloadButton.addEventListener('click', () => {
        if (countdownInterval) {
            clearInterval(countdownInterval);
        }
        detectedEmotionGif.src = "";
        detectedEmotionText.textContent = "";
        chooseEmotionText.textContent = "";
        otherEmotionsGrid.innerHTML = "";
        emotionModal.classList.remove('is-active');
        window.hasDetectionStarted = false;
        fetchDominantEmotion();
    });
    fetchDominantEmotion();
    modalBackground.addEventListener('click', () => {
        emotionModal.classList.remove('is-active');
    });
});
</script>
{% endblock %}
```

7.1.2.3 Autism/templates/activity.html

```
{% extends "base.html" %}
{% block style_content %}
<link rel="stylesheet" href="{{url_for('static',filename='css/index_3.css')}}" type="text/css">
<link rel="stylesheet" href="{{url_for('static',filename='css/aos.css')}}" type="text/css">
{% endblock %}
{% block nav_content %}
<div class="navbar-item is-flex-grow-1 is-flex is-justify-content-center">
    <div class="has-text-primary has-text-centered u--pulse">
      <p class="is-size-4 has-text-weight-semibold">Activities</p>
      <p class="is-size-6 has-text-weight-medium">Select an activity</p>
```

```
    </div>
</div>
{% endblock %}
{% block body_content %}
<div class="activity-container">
    <div class="columns is-multiline is-centered m-4">
        <div class="column is-4" data-aos="fade-up" data-aos-duration="1000">
            <a href="{{ url_for('happy_page') }}" class="box p-0 activity-card">
                <div class="activity-image">
                    <img src="{{ url_for('static', filename='images/happy_activity.png') }}"
alt="Happy Activity">
                </div>
                <div class="has-text-primary has-text-centered has-text-weight-semibold p-3">
                    Water Penguin
                </div>
            </a>
        </div>
        <div class="column is-4" data-aos="fade-up" data-aos-duration="1000" data-aos-
delay="200">
            <a href="{{ url_for('sad_page') }}" class="box p-0 activity-card">
                <div class="activity-image">
                    <img src="{{ url_for('static', filename='images/sad_activity.png') }}" alt="Sad
Activity">
                </div>
                <div class="has-text-primary has-text-centered has-text-weight-semibold p-3">
                    Breathe and Clap
                </div>
            </a>
        </div>
        <div class="column is-4" data-aos="fade-up" data-aos-duration="1000" data-aos-
delay="400">
            <a href="{{ url_for('angry_page') }}" class="box p-0 activity-card">
                <div class="activity-image">
```

```
                <img src="{{ url_for('static', filename='images/angry_activity.png') }}"
alt="Angry Activity">
            </div>
            <div class="has-text-primary has-text-centered has-text-weight-semibold p-3">
                Bubble Pop
            </div>
        </a>
    </div>
    <div class="column is-4" data-aos="fade-up" data-aos-duration="1000" data-aos-
delay="200">
        <a href="{{ url_for('surprised_page') }}" class="box p-0 activity-card">
            <div class="activity-image">
                <img src="{{ url_for('static', filename='images/surprised_activity.png') }}"
alt="Surprised Activity">
            </div>
            <div class="has-text-primary has-text-centered has-text-weight-semibold p-3">
                Sound Surprise
            </div>
        </a>
    </div>
    <div class="column is-4" data-aos="fade-up" data-aos-duration="1000" data-aos-
delay="400">
        <a href="{{ url_for('neutral_page') }}" class="box p-0 activity-card">
            <div class="activity-image">
                <img src="{{ url_for('static', filename='images/neutral_activity.png') }}"
alt="Neutral Activity">
            </div>
            <div class="has-text-primary has-text-centered has-text-weight-semibold p-3">
                Emotion Match
            </div>
        </a>
    </div>
    <div class="column is-4" data-aos="fade-up" data-aos-duration="1000" data-aos-
delay="600">
```

```
          <a href="{{ url_for('fear_page') }}" class="box p-0 activity-card">
            <div class="activity-image">
              <img src="{{ url_for('static', filename='images/fear_activity.png') }}" alt="Fear
Activity">
            </div>
            <div class="has-text-primary has-text-centered has-text-weight-semibold p-3">
              Coloring Page
            </div>
          </a>
        </div>
    </div>
</div>
{% endblock %}
{% block script_content %}
<script src="{{ url_for('static', filename='scripts/aos.js') }}"></script>
<script>
    AOS.init({
        offset: 120,
        delay: 0,
        duration: 1000,
        easing: 'ease',
        once: true,
        mirror: false,
    });
</script>
{% endblock %}
```

7.2.5 HTML Template for an Activity Page

```
{% extends "base.html" %}
{% block style_content %}
<link rel="stylesheet" href="{{url_for('static',filename='css/index_2.css')}}" type="text/css">
<style>
    h1 {
        color: white;
```

```
        margin-bottom: 10px;
    }
    canvas {
        background: black;
        display: block;
    }
</style>
{% endblock %}
{% block nav_content %}
<div class="navbar-item is-flex-grow-1 is-flex is-justify-content-center">
    <div class="has-text-primary has-text-centered u--pulse">
      <p class="is-size-4 has-text-weight-semibold">Activity</p>
      <p class="is-size-6 has-text-weight-medium">Bubble Pop</p>
    </div>
</div>
{% endblock %}
{% block body_content %}
<audio id="bgMusic" loop autoplay>
    <source src="{{ url_for('static', filename='audio/angry/Fuzz Buzz - Jeremy Korpas.mp3')
}}" type="audio/mp3">
</audio>
<div class="game-container">
    <canvas id="gameCanvas" class="u--bounce"></canvas>
</div>
{% endblock %}
{% block script_content %}
<script src="{{url_for('static', filename='scripts/js-confetti.browser.js')}}"></script>
<script>
    document.addEventListener("DOMContentLoaded", function() {
        const bgMusic = document.getElementById("bgMusic");
        bgMusic.volume = 0.3;
        const playPromise = bgMusic.play();
        if (playPromise !== undefined) {
            playPromise.catch(() => {
```

```
            document.addEventListener("click", () => {
                bgMusic.play();
            }, { once: true });
        });
    }
});
 function resizeCanvas() {
    const canvas = document.getElementById("gameCanvas");
    const ctx = canvas.getContext("2d");
    canvas.width = window.innerWidth * 0.9;
    canvas.height = window.innerHeight * 0.75;
}
window.addEventListener("load", resizeCanvas);
window.addEventListener("resize", resizeCanvas);
const BACKGROUND_IMAGE_URL = "{{ url_for('static',
filename='activity/A3/background.jpg') }}";
const BUBBLE_IMAGE_URL = "{{ url_for('static', filename='activity/A3/bubble.png')
}}";
const POP_SOUND_URL = "{{ url_for('static', filename='activity/A3/Pop.mp3') }}";
const ENTRY_SOUND_URL = "{{ url_for('static', filename='activity/A3/Breeze
Entry.mp3') }}";
const APPLAUSE_SOUND_URL = "{{ url_for('static', filename='audio/clap.mp3') }}";
</script>
<script src="{{ url_for('static', filename='activity/A3/script.js') }}"></script>
{% endblock %}
```

7.2.6 HTML Template for a Music Page

```
{% extends "base.html" %}
{% block style_content %}
<link rel="stylesheet" href="{{ url_for('static', filename='css/index_3.css') }}"
type="text/css">
<link rel="stylesheet" href="{{ url_for('static', filename='css/aos.css') }}" type="text/css">
<style>
    .music-play-btn {
```

```
    background: none;
    border: none;
    cursor: pointer;
    padding: 0.5rem;
    margin-left: 1rem;
    transition: transform 0.2s ease;
    display: inline-flex;
    align-items: center;
    vertical-align: middle;
}
.music-play-btn svg {
    fill: hsl(171, 100%, 41%);
    width: 32px;
    height: 32px;
}
.music-text-container {
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 0.5rem;
}
.music-play-btn:hover {
    transform: scale(1.1);
}
.music-play-btn:focus {
    outline: none;
}
.music-grid {
    width: 100%;
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 20px;
    display: flex;
    flex-wrap: wrap;
```

```
        justify-content: center;
        overflow-y: unset;
        box-sizing: border-box;
    }
    .music-card-wrapper {
        flex: 0 0 calc(33.333% - 40px);
        margin: 20px;
        box-sizing: border-box;
    }
    @media (max-width: 768px) {
        .music-card-wrapper {
            flex: 0 0 calc(100% - 40px);
        }
    }
</style>
{% endblock %}
{% block nav_content %}
<div class="navbar-item is-flex-grow-1 is-flex is-justify-content-center">
    <div class="has-text-primary has-text-centered u--pulse">
        <p class="is-size-4 has-text-weight-semibold">Angry Music</p>
        <p class="is-size-6 has-text-weight-medium">Select a song to play</p>
    </div>
</div>
{% endblock %}
{% block body_content %}
<div style="display: none;">
    <svg id="pause-icon" xmlns="http://www.w3.org/2000/svg">
        <path d="M12 2a10 10 0 1 0 10 10A10 10 0 0 0 12 2zm0 18a8 8 0 1 1 8-8 8 8 0 0 1-8
8z"/>
        <path d="M15 8a1 1 0 0 0-1 1v6a1 1 0 0 0 2 0V9a1 1 0 0 0-1-1z"/>
        <path d="M9 8a1 1 0 0 0-1 1v6a1 1 0 0 0 2 0V9a1 1 0 0 0-1-1z"/>
    </svg>
    <svg id="play-icon" xmlns="http://www.w3.org/2000/svg">
```

```
        <path d="M12 2a10 10 0 1 0 10 10A10 10 0 0 0 12 2zm0 18a8 8 0 1 1 8-8 8 8 0 0 1-8 8z"/>
        <path d="M12.34 7.45a1.7 1.7 0 0 0-1.85-.3 1.6 1.6 0 0 0-1 1.48v6.74a1.6 1.6 0 0 0 1 1.48 1.68 1.68 0 0 0 .69.15 1.74 1.74 0 0 0 1.16-.45L16 13.18a1.6 1.6 0 0 0 0-2.36zm-.84 7.15V9.4l2.81 2.6z"/>
    </svg>
</div>
<div class="music-container">
    <div class="music-grid">
        <div class="music-card-wrapper" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="100">
            <div class="box p-0 music-card">
                <div class="music-image">
                    <img src="{{ url_for('static', filename='images/angry_art.jpg') }}" alt="Angry Music">
                </div>
                <div class="music-text-container has-text-weight-semibold p-3">
                    Exploring
                    <button class="music-play-btn" onclick="togglePlay(this, 'Exploring')">
                        <svg class="play-icon" width="32" height="32"></svg>
                    </button>
                </div>
                <audio id="Exploring" src="{{ url_for('static', filename='audio/angry/Exploring - Jeremy Korpas.mp3') }}"></audio>
            </div>
        </div>
        <div class="music-card-wrapper" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="200">
            <div class="box p-0 music-card">
                <div class="music-image">
                    <img src="{{ url_for('static', filename='images/angry_art.jpg') }}" alt="Angry Music">
                </div>
                <div class="music-text-container has-text-weight-semibold p-3">
```

```
                Fuzz Buzz
                <button class="music-play-btn" onclick="togglePlay(this, 'Fuzz Buzz')">
                    <svg class="play-icon" width="32" height="32"></svg>
                </button>
            </div>
            <audio id="Fuzz Buzz" src="{{ url_for('static', filename='audio/angry/Fuzz Buzz - Jeremy Korpas.mp3') }}"></audio>
        </div>
    </div>
    <div class="music-card-wrapper" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="300">
        <div class="box p-0 music-card">
            <div class="music-image">
                <img src="{{ url_for('static', filename='images/angry_art.jpg') }}" alt="Angry Music">
            </div>
            <div class="music-text-container has-text-weight-semibold p-3">
                Heat Warning
                <button class="music-play-btn" onclick="togglePlay(this, 'Heat Warning')">
                    <svg class="play-icon" width="32" height="32"></svg>
                </button>
            </div>
            <audio id="Heat Warning" src="{{ url_for('static', filename='audio/angry/Heat Warning - Jeremy Korpas.mp3') }}"></audio>
        </div>
    </div>
    <div class="music-card-wrapper" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="400">
        <div class="box p-0 music-card">
            <div class="music-image">
                <img src="{{ url_for('static', filename='images/angry_art.jpg') }}" alt="Angry Music">
            </div>
            <div class="music-text-container has-text-weight-semibold p-3">
```

```
                Test Run
                <button class="music-play-btn" onclick="togglePlay(this, 'Test Run')">
                    <svg class="play-icon" width="32" height="32"></svg>
                </button>
            </div>
            <audio id="Test Run" src="{{ url_for('static', filename='audio/angry/Test Run - Jeremy Korpas.mp3') }}"></audio>
        </div>
    </div>
    <div class="music-card-wrapper" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="500">
        <div class="box p-0 music-card">
            <div class="music-image">
                <img src="{{ url_for('static', filename='images/angry_art.jpg') }}" alt="Angry Music">
            </div>
            <div class="music-text-container has-text-weight-semibold p-3">
                The Grime
                <button class="music-play-btn" onclick="togglePlay(this, 'The Grime')">
                    <svg class="play-icon" width="32" height="32"></svg>
                </button>
            </div>
            <audio id="The Grime" src="{{ url_for('static', filename='audio/angry/The Grime - Jeremy Korpas.mp3') }}"></audio>
        </div>
    </div>
    <div class="music-card-wrapper" data-aos="fade-up" data-aos-duration="1000" data-aos-delay="600">
        <div class="box p-0 music-card">
            <div class="music-image">
                <img src="{{ url_for('static', filename='images/angry_art.jpg') }}" alt="Angry Music">
            </div>
            <div class="music-text-container has-text-weight-semibold p-3">
```

```
                Tough Times
                <button class="music-play-btn" onclick="togglePlay(this, 'Tough Times')">
                    <svg class="play-icon" width="32" height="32"></svg>
                </button>
            </div>
            <audio id="Tough Times" src="{{ url_for('static', filename='audio/angry/Tough
Times - Jeremy Korpas.mp3') }}"></audio>
        </div>
    </div>
  </div>
</div>
{% endblock %}
{% block script_content %}
<script src="{{ url_for('static', filename='scripts/aos.js') }}"></script>
<script>
  AOS.init({
    offset: 120,
    delay: 0,
    duration: 1000,
    easing: 'ease',
    once: true,
    mirror: false,
  });
  let currentlyPlaying = null;
  let currentButton = null;
  const playIconTemplate = document.getElementById('play-icon').outerHTML;
  const pauseIconTemplate = document.getElementById('pause-icon').outerHTML;
  document.addEventListener('DOMContentLoaded', function() {
    document.querySelectorAll('.music-play-btn svg').forEach(icon => {
      icon.outerHTML = playIconTemplate;
    });
  });
  function togglePlay(button, audioId) {
    const audio = document.getElementById(audioId);
```

```
        const icon = button.querySelector('svg');
        if (currentlyPlaying && currentlyPlaying !== audio) {
            currentlyPlaying.pause();
            currentlyPlaying.currentTime = 0;
            if (currentButton) {
                currentButton.querySelector('svg').outerHTML = playIconTemplate;
            }
        }
        if (audio.paused) {
            audio.play();
            icon.outerHTML = pauseIconTemplate;
            currentlyPlaying = audio;
            currentButton = button;
        } else {
            audio.pause();
            icon.outerHTML = playIconTemplate;
            currentlyPlaying = null;
            currentButton = null;
        }
    }
</script>
{% endblock %}
```

7.2.7 Primary CSS Stylesheet

```
body{
    background-color: #fff;
}
::selection {
  background-color: hsl(171, 100%, 41%);
  color: white;
}
::-moz-selection {
  background-color: hsl(171, 100%, 41%);
  color: white;
```

```
}
::-webkit-scrollbar {
  width: 12px;
}
::-webkit-scrollbar-track {
  background: #f1f1f1;
}
::-webkit-scrollbar-thumb {
  background: hsl(171, 100%, 41%);
  border-radius: 6px;
}
::-webkit-scrollbar-thumb:hover {
  background: hsl(171, 100%, 35%);
}
.navbar{
  display: none;
}
.navbar-menu{
                        box-shadow: none;
}
.video-container{
  position: absolute;
  top: calc(var(--navbar-height, 80px) + 20px);
  left: 0;
  right: 0;
  bottom: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  background-image: url("../images/bg.jpg");
  background-size: cover;
  background-position: center;
}
.video-container{
```

```
  display: none;
}
.video-cam {
  max-height: calc(100% - 40px);
  max-width: calc(100% - 40px);
  width: auto;
  height: auto;
  border-radius: 10px;
}
#bgAudio{
                        display: none;
}
.u--flash-1{
  animation-delay:0.1s;
}
.u--flash-1{
  animation-delay:0.2s;
}
.u--flash-1{
  animation-delay:0.3s;
}
.loading-screen {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    background-color: #f5f5f5;
    background-image: url("bg.jpg");
    background-size: cover;
    background-position: center;
}
.logo {
    width: 40vw;
```

```
    max-width: 200px;
    min-width: 100px;
    margin-bottom: 20px;
}
.lds-ellipsis,
.lds-ellipsis div {
  box-sizing: border-box;
}
.lds-ellipsis {
  display: inline-block;
  position: relative;
  width: 80px;
  height: 80px;
}
.lds-ellipsis div {
  position: absolute;
  top: 33.33333px;
  width: 13.33333px;
  height: 13.33333px;
  border-radius: 50%;
  background: #00d1b2;
  animation-timing-function: cubic-bezier(0, 1, 1, 0);
}
.lds-ellipsis div:nth-child(1) {
  left: 8px;
  animation: lds-ellipsis1 0.6s infinite;
}
.lds-ellipsis div:nth-child(2) {
  left: 8px;
  animation: lds-ellipsis2 0.6s infinite;
}
.lds-ellipsis div:nth-child(3) {
  left: 32px;
  animation: lds-ellipsis2 0.6s infinite;
```

```
}
.lds-ellipsis div:nth-child(4) {
  left: 56px;
  animation: lds-ellipsis3 0.6s infinite;
}
@keyframes lds-ellipsis1 {
  0% {
    transform: scale(0);
  }
  100% {
    transform: scale(1);
  }
}
@keyframes lds-ellipsis3 {
  0% {
    transform: scale(1);
  }
  100% {
    transform: scale(0);
  }
}
@keyframes lds-ellipsis2 {
  0% {
    transform: translate(0, 0);
  }
  100% {
    transform: translate(24px, 0);
  }
}
.modal-content {
    background: white;
    padding: 20px;
    border-radius: 10px;
    width: 70vw;
```

```
    max-width: 800px;
    text-align: center;
    margin: 0 auto;
    overflow-y: auto;
    max-height: 90vh;
}
.box {
    padding: 20px;
    border-radius: 10px;
}
#detectedEmotionGif {
    max-width: 100px;
    height: auto;
}
.emotion-choice {
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    border-radius: 8px;
    padding: 8px;
    transition: box-shadow 0.2s ease-in-out;
    text-align: center;
}
.emotion-choice:hover {
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}
#otherEmotionsGrid {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 10px;
}
#otherEmotionsGrid img {
    max-width: 40px;
    height: auto;
}
```

```
#otherEmotionsGrid p {
    font-size: 0.9rem;
    margin-top: 0.5rem;
}
#countdown {
    color: #00d1b2;
    font-weight: bold;
}
#countdown-number {
    color: #ff3860;
    font-size: 1.2em;
}
.modal {
    padding: 10px;
}
#otherEmotionsGrid {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 10px;
    justify-items: stretch;
    align-items: stretch;
    width: 100%;
    max-width: 600px;
    margin: 0 auto;
}
.emotion-choice {
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    border-radius: 8px;
    padding: 16px;
    transition: box-shadow 0.2s ease-in-out;
    text-align: center;
    display: flex;
    flex-direction: column;
    align-items: center;
```

```
    justify-content: center;
    min-width: 100px;
    min-height: 100px;
    width: 100%;
    height: 100%;
}
.emotion-choice:hover {
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}
#otherEmotionsGrid img {
    max-width: 40px;
    height: auto;
}
#otherEmotionsGrid p {
    font-size: 0.9rem;
    margin-top: 0.5rem;
}
@media (max-width: 768px) {
    #otherEmotionsGrid {
        grid-template-columns: repeat(2, 1fr);
    }
}
@media (max-width: 480px) {
    #otherEmotionsGrid {
        grid-template-columns: 1fr;
    }
}
```

7.2.8 Interactive JavaScript Activity Example

```
document.addEventListener('DOMContentLoaded', function() {
    const jsConfetti = new JSConfetti();
    const button = document.getElementById('drinkButton');
    const svg = document.querySelector('svg');
    const sipCounter = document.getElementById('sipCounter');
```

```
button.textContent = 'Take a Sip!';
const quackSound = new Audio(QUACK_SOUND_URL);
const yaySound = new Audio(YAY_SOUND_URL);
const applauseSound = new Audio(APPLAUSE_SOUND_URL);
let sipCount = 0;
const totalSips = 5;
function resetGame() {
    sipCount = 0;
    button.disabled = false;
    button.textContent = 'Take a Sip!';
    sipCounter.textContent = `Sips: ${sipCount}/${totalSips}`;
    const rect = document.querySelector('#fullClip rect');
    rect.setAttribute('y', '2650');
}
svg.addEventListener('animationend', () => {
    svg.classList.remove('wiggle-animation');
});
button.addEventListener('click', function() {
    if (sipCount >= totalSips) {
        resetGame();
        return;
    }
    if (sipCount < totalSips - 1) {
        quackSound.currentTime = 0;
        quackSound.play();
    }
    svg.classList.remove('wiggle-animation');
    void svg.offsetWidth;
    svg.classList.add('wiggle-animation');
    const rect = document.querySelector('#fullClip rect');
    const startY = 2650;
    const endY = 500;
    const totalDistance = startY - endY;
    const distancePerSip = totalDistance / totalSips;
```

```
        sipCount++;
        sipCounter.textContent = `Sips: ${sipCount}/${totalSips}`;
        const targetY = startY - (distancePerSip * sipCount);
        const duration = 1000;
        const startTime = performance.now();
        const animationStartY = parseInt(rect.getAttribute('y'));
        function animate(currentTime) {
            const elapsed = currentTime - startTime;
            const progress = Math.min(elapsed / duration, 1);
            const newY = animationStartY - (animationStartY - targetY) * progress;
            rect.setAttribute('y', newY);
            if (progress < 1) {
                requestAnimationFrame(animate);
            }
        }
        requestAnimationFrame(animate);
        if (sipCount >= totalSips) {
            button.textContent = 'Take a Drink Again!';
            yaySound.currentTime = 0;
            yaySound.play();
            applauseSound.currentTime = 0;
            applauseSound.play();
            jsConfetti.addConfetti({
                confettiNumber: 500
            });
        }
    });
});
```

8. RESULTS AND DISCUSSION

8.1 EVALUATION MEASURES

Evaluation techniques are methods used to measure the performance of a machine learning model by determining how well it makes correct predictions. They assist in establishing how well a model can generalize to new data, where improvements are needed, and how models can be compared to choose the best one. In our project, we use accuracy and loss curves and confusion matrix to measure the performance of the emotion recognition model.

8.1.1 Accuracy Curve and Loss Curve

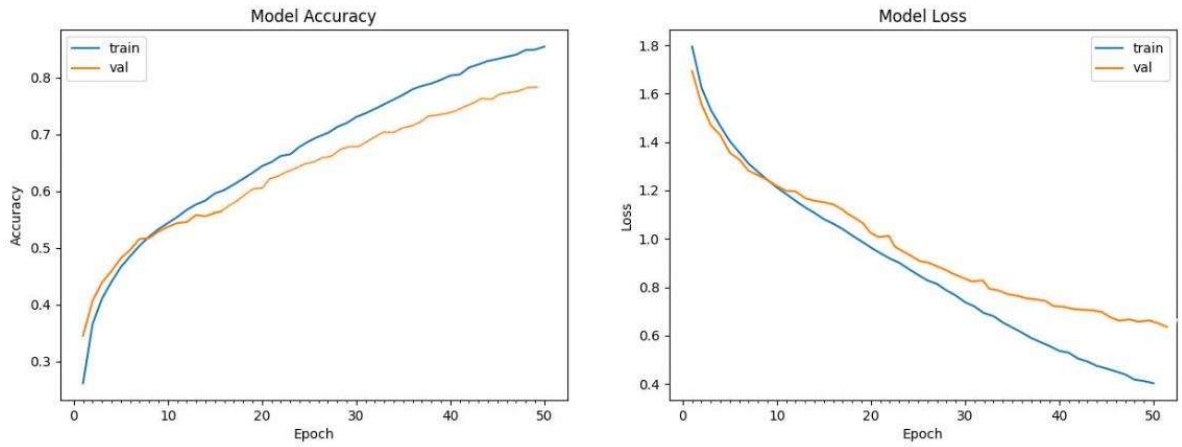

Figure 6 Accuracy and Loss Curve Diagram

Accuracy and loss curves are graphs that are used to monitor a model's performance while training. The accuracy curve indicates the capability of the model to classify data across epochs, whereas the loss curve represents the reduction in errors, which assists in detecting problems such as overfitting or underfitting. The curves in Figure 6 indicate the training process of the EmoCare's emotion classification model, which had been trained for 50 epochs. The graph on the left is the accuracy graph, which depicts the increasing performance of the model across epochs as both the training and validation accuracy rises gradually. Although the training accuracy rises over 80%, the validation accuracy shows a slight lag, suggesting a gap arising from issues of generalization. The loss graph on the right indicates a steady drop in both training and validation loss, which implies that the model is learning well.

But the slight deviation between training and validation loss in subsequent epochs could be an indication of some overfitting.

8.1.2 Confusion Matrix

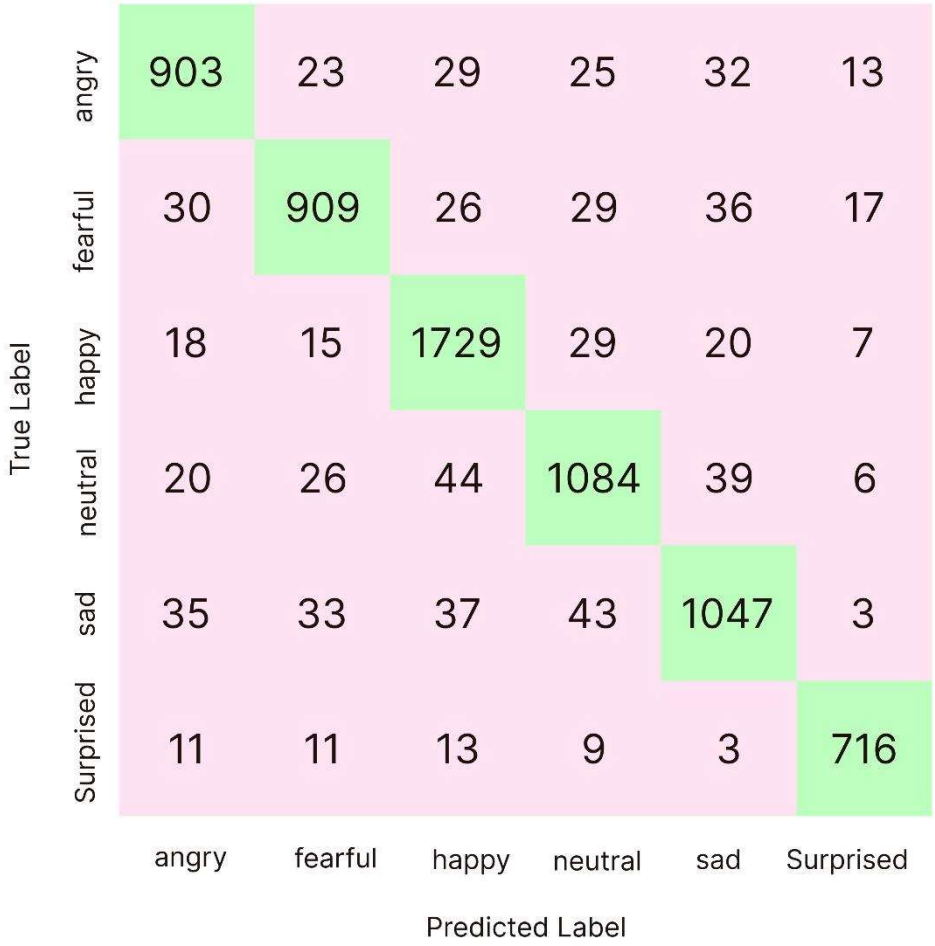

Figure 7 Confusion Matrix

A confusion matrix is a matrix used to describe the performance of a classification model, with actual and predicted labels. It can be used to analyze misclassifications, and it provides insights into precision, recall, and accuracy. The confusion matrix in Figure 7 shows how well the EmoCare emotion recognition model classifies six emotions: angry, fearful, happy, neutral, sad, and surprised. The correctly classified instances are shown along the diagonal, with "happy" being the best accuracy at 1,729 correct predictions. Other emotions, like "angry" (903) and "neutral" (1,084), also show good

classification performance. Misclassifications where there are non-diagonal values are as follows. For instance, "sad" is at times confused as "neutral" (43 occurrences) or "happy" (37 occurrences). In the same way, "fearful" is also confused with "angry" (30 occurrences) and "sad" (36 occurrences). The comparatively few misclassifications indicate that overall the model performs well. A comprehensive analysis of the performance of the model is also seen through the major metrics of precision, recall, and F1-score, which offer greater insights into its performance for various emotion classes.

Below are the formulas for calculating thr queries:

$$\text{Accuracy} = \frac{TP+TN}{TP+FN+TN+FP}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\text{F1-Score} = \frac{2\times(\text{Precision}\times\text{Recall})}{\text{Precision}+\text{Recall}}$$

The calculated metrics for the EmoCare model are:

Accuracy: 93.4%

Precision: 90.35%

Recall: 90.03%

F1-Score: 90.18%

9. SYSTEM TESTING

System testing is vital in maintaining the quality and reliability of EmoCare. During this stage, the system is tested as an entity confirming that all components are working properly interact smoothly and provide expected results. The testing process ensures that EmoCare accurately detects emotions records video efficiently and provides a smooth user experience across various devices and environments.

9.1 LEVELS OF TESTING

9.1.1 Unit Testing

To confirm the precision and dependability of each component of EmoCare unit testing was done. Each module was tested independently to ensure it performed as expected. The face detection module was evaluated to confirm that the system accurately detects a face before initiating recording ensuring it does not start unnecessarily or miss valid faces. The recording module was also assessed to verify that video recording starts and stops correctly based on face detection triggers preventing unnecessary memory usage and ensuring proper data capture.

9.1.2 System Testing

System testing assessed the overall functionality of EmoCare ensuring all integrated components worked cohesively. Functional testing was performed to confirm that core features such as emotion detection video recording visual schedules and interactive activities functioned as intended with each activity tested for responsiveness accuracy and expected user interactions. Performance testing measured system response time and latency in emotion detection to ensure a seamless experience without lags or delays. To ensure a consistent user experience compatibility testing assessed the system across various devices browsers and screen sizes.

9.1.3 Validation Testing

Validation testing was performed to confirm that EmoCare handles both valid and invalid inputs appropriately. The system was tested using clear face images real-time user expressions and expected activity interactions to ensure correct detections and responses.

Invalid inputs such as blurry images multiple faces in a frame or incomplete interactions were also introduced and the system was expected to provide appropriate error messages or fallback mechanisms to handle unexpected conditions. Additionally specific activities were tested to verify expected behaviors. The visual schedule was tested to ensure that tasks were updated upon selection remained persistent after page refresh and were marked as completed when clicked. Music and sound features were checked to verify that the play pause and resume functions operated correctly ensuring seamless transitions between tracks. Activity background music was tested to confirm that it played consistently across all activities and stopped appropriately when required. The bubble pop activity was assessed to ensure that a single bubble appeared at a time with the next one only appearing after the previous one was popped. The penguin game was tested to confirm that the level reset after completing five consecutive sips. The randomness in activities was also verified ensuring that activities incorporating randomized elements such as emotion matching and surprise sound effects behaved as expected with varied outputs in different sessions. UI and responsiveness were also tested to ensure proper display and usability across desktops tablets and mobile devices.

9.2 TEST CASE

Test ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status
Emotion Detection & Recording						
1	Face Detection	1. Open EmoCare. 2. Position face within the frame.	Valid face image	Detects face and starts recording	Detects face and starts recording	Pass
2	Invalid Face Detection	1. Open EmoCare. 2. Show a blurry face.	Blurry image	doesn't start the recording	doesn't start the recording	Pass
3	Emotion Recognition	1. Show a happy face. 2. Capture frames for 10 sec.	Image with happy expression	Detects "Happy"	Detects "Happy"	Pass
Breath & Clap Activity						
4	Breath & Clap Animation Loop	1. Start the exercise. 2. Display the animation repetitively	Valid user actions	System detects actions correctly	System detects actions correctly	Pass
Coloring Page Activity						
5	Coloring Interaction	1. Click a region in the SVG.	Valid coloring actions	Region fills with color,	Works as expected	Pass

		2. Complete coloring.		new SVG appears		
Emotion Matching Activity						
6	Correct Selection	1. Show an emoji. 2. Select the correct emoji.	Emoji selection	Progresses to the next emoji	Works as expected	Pass
7	Incorrect Selection	1. Select the wrong emoji.	Incorrect emoji	Plays "wrong" sound & shakes	Works as expected	Pass
Water Penguin Activity						
8	Water Tracking	1. Click on 'Take a Sip Button'. 2. Check tummy fill.	Button Clicks	Penguin's tummy fills accordingly	Works as expected	Pass
9	Level Reset	1. Complete 5 consecutive sips.	5 Button Clicks	Level resets	Level resets	Pass
Visual Schedule						
10	Updating To-Do List	1. Click a task from available tasks.	Task selection	Moves task to "To-Do" list	Works as expected	Pass
11	Task Persistence	1. Refresh the page. 2. Check To-Do list.	Page reload	Tasks remain in "To-Do" list	Works as expected	Pass
12	Task Completion	1. Click a task in the "To-Do" list.	Task marked complete	Task disappears from the list	Works as expected	Pass
Music & Sound						
13	Music Play Functionality	1. Click play on a track.	Click action	Music starts playing	Music starts playing	Pass
14	Pause & Resume Music	1. Click pause. 2. Click play again.	Click action	Music resumes from paused position	Works as expected	Pass
15	Switching Tracks	1. Play a track. 2. Click play on another track.	Multiple tracks	New track starts, previous stops	Works as expected	Pass
UI & Responsiveness						
16	Emotion Loading Check	1. Open the application.	System start	Emotion model loads successfully	Works as expected	Pass
17	Responsive Design	1. Resize browser to different sizes.	Various screen sizes	Layout adjusts correctly	Works as expected	Pass

18	Developer Console Errors	1. Open Developer Tools. 2. Load each activity.	System elements	No missing elements, no errors	Works as expected	Pass
Sound Surprise Activity						
19	Sound Surprise Buttons	1. Click every sound button.	Click action	Each button produces sound	Works as expected	Pass
Bubble Pop Activity						
20	Bubble Pop - Sequential Bubble Appearance	1. Click on the bubble to pop it. 2. Observe if the next bubble appears only after the previous one is popped.	Click action	Only one bubble appears at a time, and the next one appears only after popping the current bubble	Works as expected	Pass
Audio Auto-Play (Browser Permissions)						
21	Activity Scoreboard	1. Complete an activity. 2. Check scoreboard.	Activity results	Score updates correctly	Works as expected	Pass
22	Welcome Audio Auto-Play	1. Load the application. 2. Allow audio in browser.	Auto-play test	Welcome audio plays automatically	Works as expected	Pass
23	Activity Background Music Auto-Play	1. Start an activity. 2. Allow auto-play.	Auto-play test	Background music starts automatically	Works as expected	Pass
Randomness in Activity						
24	Randomness Verification in Activities	1. Start activities with randomized elements. 2. Restart the activity multiple times. 3. Observe if elements change as expected.	Activity start/restart action	Randomized elements should change on each attempt	Works as expected	Pass

Table 2 Test Cases

10. CONCLUSION

In summary, the ground-breaking web application EmoCare was created to promote the emotional health of kids with autism. Through the utilization of sophisticated machine learning methodologies and the integration of a wide variety of emotive activities EmoCare has become a comprehensive platform that caters to the unique emotional requirements of its users. EmoCare offers a tailored experience that fosters emotional intelligence self-regulation and general well-being thanks to its user-friendly features and intuitive interface. The application's many modules which include soothing coloring pages guided breathing exercises and an emotion-matching activity provide entertaining and instructive experiences that promote normal emotional development. The significance of preserving both physical and mental well-being is emphasized by the addition of a Drinking Water Reminder and relaxing music playlists. Children can also participate in stress relief and routine creation through the Bubble Pop activity and Visual Schedule features respectively. With an emphasis on the emotional needs of kids with autism, EmoCare's holistic approach seeks to establish a safe accepting environment that fosters emotional growth and self-awareness. By emphasizing successful results and user experiences EmoCare aims to improve user’s emotional wellbeing by having a positive impact on their lives. To sum up, EmoCare is a state-of-the-art emotional support platform created to empower kids with autism not just an application. It is an invaluable resource for parents, caregivers, and educators looking to support young people's emotional development because of its cutting-edge features user-friendly interface, and all-encompassing approach.

11. FUTURE ENHANCEMENTS

Scaling this project offers a number of areas for improvement and future development. One of the most important areas of development is the addition of more activities, music and visual schedule tasks, to provide a more engaging experience for children. Another area for development is expanding the model with more images and a greater diversity of images, which can dramatically improve its ability to recognize emotions. Future developments will involve enhancing the responsiveness of the system to tailor itself more to the special needs of individual users as well as incorporating real-time feedback loops to enhance its interactive value. Improving the user interface to make it more seamless and intuitive would further enhance accessibility and usability for children. Ongoing refinement through user feedback and evolving emotion recognition techniques will maintain the system's utility and influence in assisting children with ASD.

12. PUBLICATIONS

Flickson, J., Fiza Fariya, Mohammed Nihal, & Anandhu, J.,2025, “Emocare: Emotional well-being app for kids with autism”, International Conference on Advanced Information Science and Computing Systems, PP:7–8

13. REFERENCES

[1] N. Gagana, K. C. Kavya, and H. Shetty, "Human emotion detection using machine learning," *International Journal of Research in Applied Science and Engineering Technology*, vol. 11, pp. 3065-3070, 2023, doi: 10.22214/ijraset.2023.51732.

[2] H.-J. Lee and K.-S. Hong, "A study on emotion recognition method and its application using face image," in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, 2017, doi: 10.1109/ictc.2017.8191005.

[3] R. S. Deshmukh, V. Jagtap, and S. Paygude, "Facial emotion recognition system through machine learning approach," in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2017, doi: 10.1109/iccons.2017.8250725.

[4] A. Singh and A. Maurya, "Human emotion detection using machine learning," *International Journal of Research in Applied Science and Engineering Technology*, vol. 10, no. V, pp. 3335-3339, 2022, doi: 10.22214/ijraset.2022.4310.

[5] A. Wutich, A. Brewis, and A. Tsai, "Water and mental health," *WIREs Water*, vol. 7, no. e1461, 2020, doi: 10.1002/wat2.1461.

[6] R. Puri, A. Gupta, M. Sikri, M. Tiwari, N. Pathak, and S. Goel, "Emotion detection using image processing in Python," *arXiv*, pp. 1389-1395, 2020, doi: 10.48550/arXiv.2012.00659.

[7] V. Cavioni, I. Grazzani, and V. Ornaghi, "Social and emotional learning for children with learning disability: Implications for inclusion," *International Journal of Emotional Education*, vol. 9, no. 2, pp. 100-109, 2017.

[8] C. Papoutsi and A. Drigas, "Mobile applications to improve emotional intelligence in autism – A review," *iJIM*, vol. 12, no. 6, pp. 47-61, 2018, doi: 10.3991/ijim.v12i6.9073.

[9] J. M. Garcia-Garcia, V. M. R. Penichet, and M. D. Lozano, "Using emotion recognition technologies to teach children with autism spectrum disorder how to identify and express emotions," *Universal Access in the Information Society*, vol. 21, pp. 809-825, 2022, doi: 10.1007/s10209-021-00818-y.

[10] M. Somerton, "Developing an educational app for students with autism," *Frontiers in Education*, vol. 7, 2022, doi: 10.3389/feduc.2022.998694.

[11] T. Maimunah, N. Yuliati, and S. Saputri, "The analysis of emotional development in autistic children age 4-5 years," in *Proceedings of the 5th International Conference on Humanities and Social Science Research (ICHSSR)*, 2020, doi: 10.2991/assehr.k.201112.046.

[12] Y. Li and A. S. Elmaghraby, "A framework for using games for behavioral analysis of autistic children," in *2014 Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES)*, 2014, doi: 10.1109/cgames.2014.6934157.

[13] S. Iyer and D. R. Kalbande, "Research on educative games for autistic children," in *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, 2014, doi: 10.1109/cscita.2014.6839296.

[14] N. Heni and H. Hamam, "Design of emotional educational system mobile games for autistic children," in *2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2016, doi: 10.1109/atsip.2016.7523168.

[15] F. M. Talaat, Z. H. Ali, R. R. Mostafa, *et al.*, "Real-time facial emotion recognition model based on kernel autoencoder and convolutional neural network for autism children," *Soft Computing*, vol. 28, pp. 6695–6708, 2024. doi: 10.1007/s00500-023-09477-y.

[16] F. M. Alamgir, S. M. H. Saif, S. M. Hossain, A. A. Hadi, and M. S. Alam, "Article title," *European Chemical Bulletin*, 2023. doi:10.48047/ecb/2023.12.Si4.1851.

14. APPENDIX

14.1 UI SCREENSHOTS

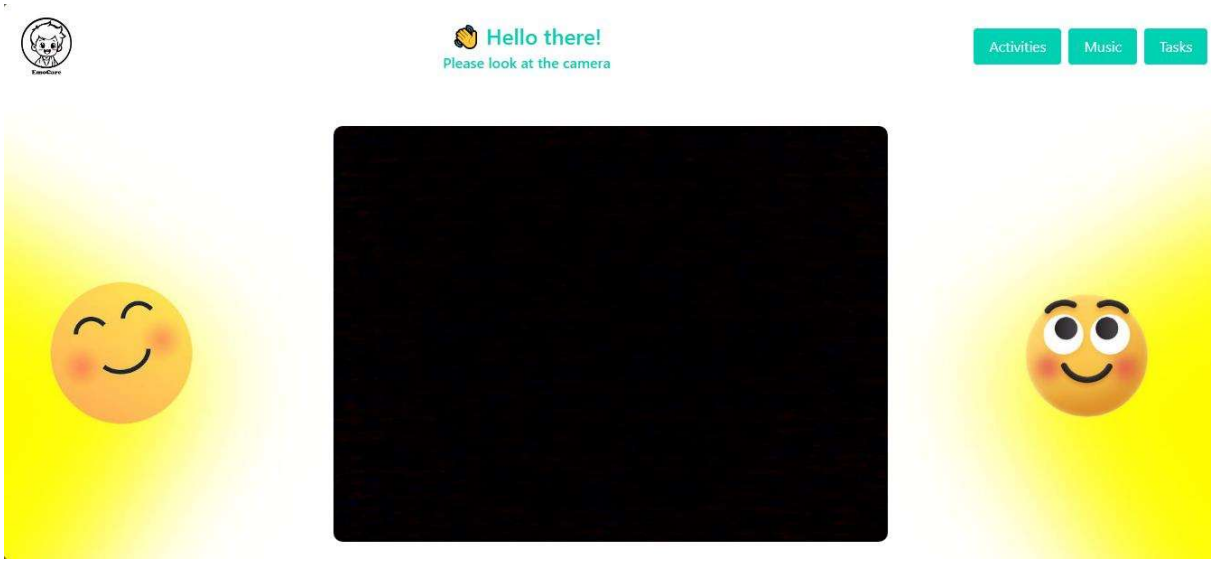

Figure 8 UI – Video Capture Page

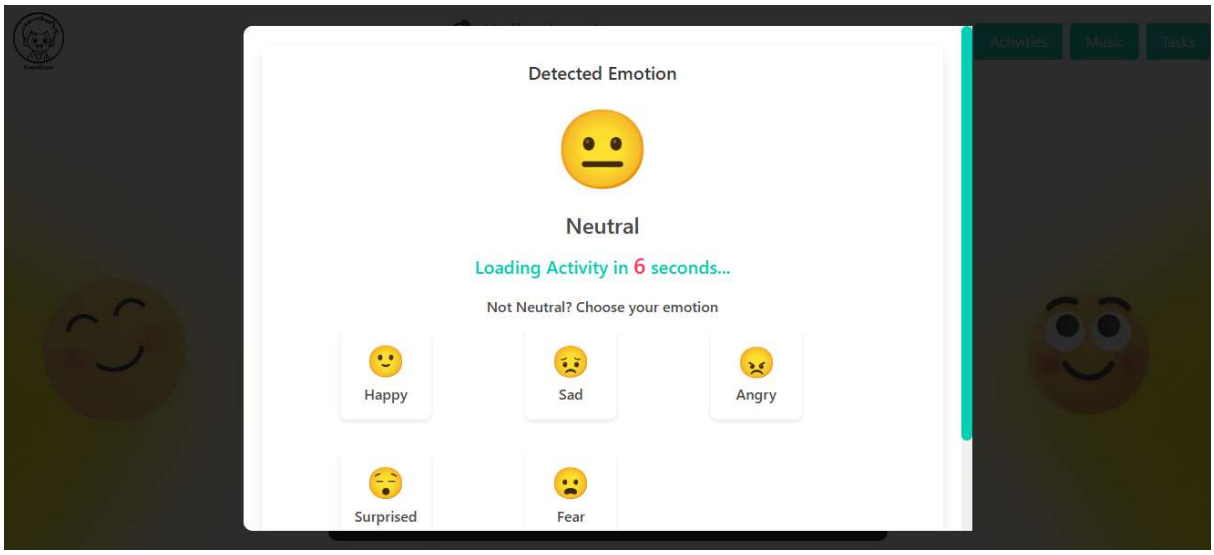

Figure 9 UI – Visual Cues Pop-Up

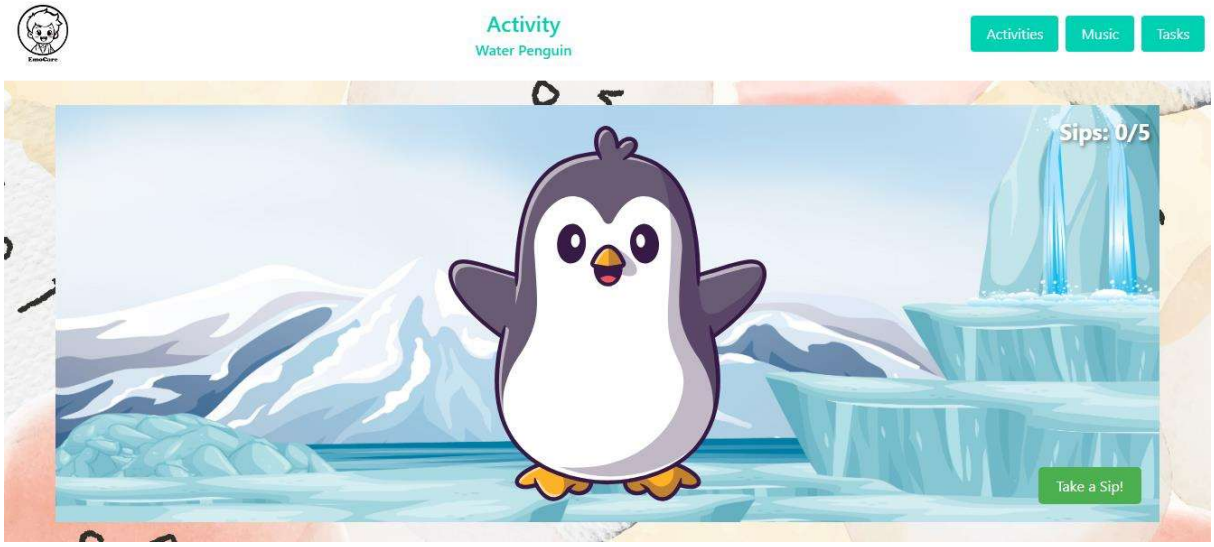

Figure 10 UI – Water Penguin Activity

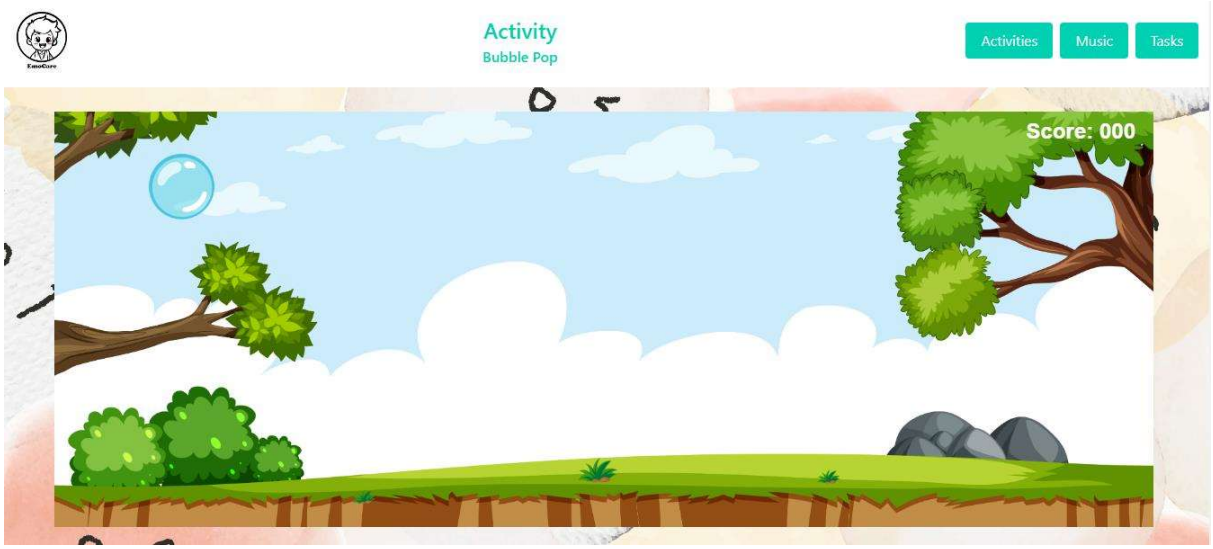

Figure 11 UI – Bubble Pop Activity

Figure 12 UI – Sound Surprise Activity

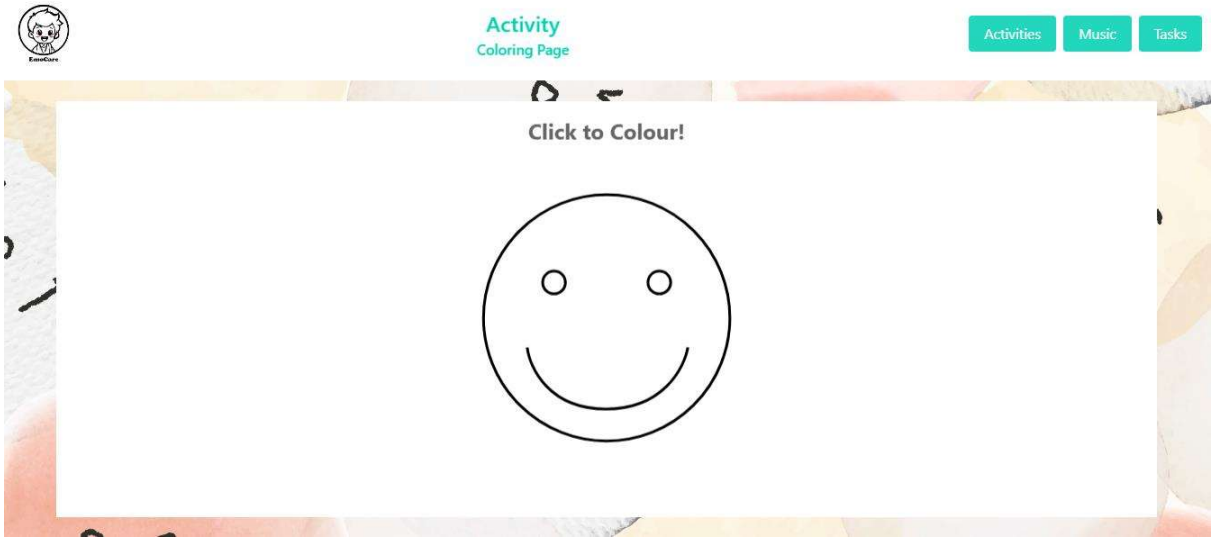

Figure 13 UI – Colouring Page Activity

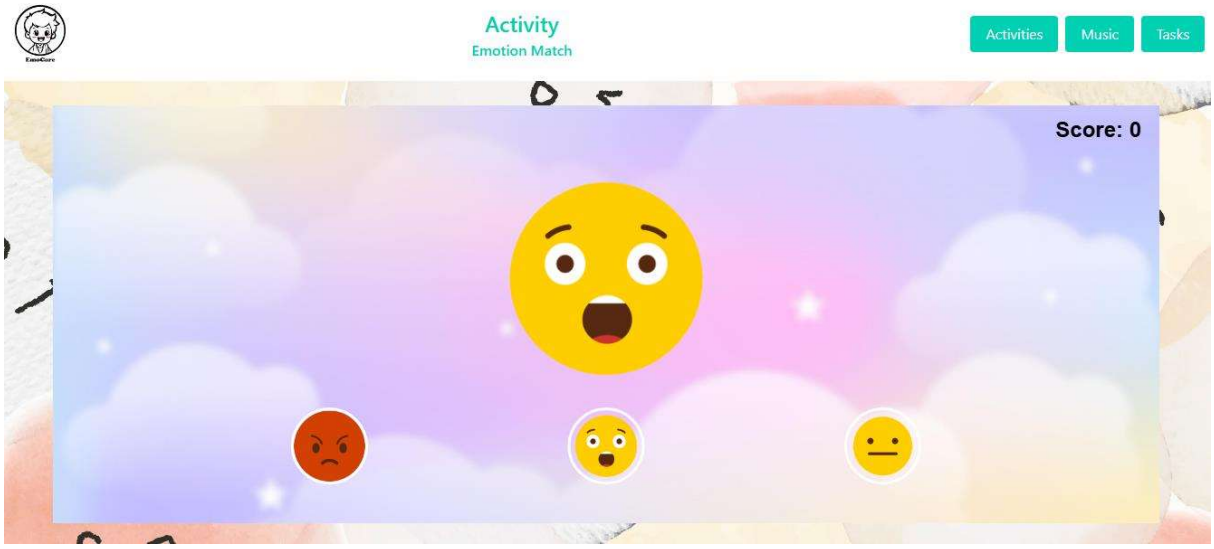

Figure 14 UI - Emotion Match Activity

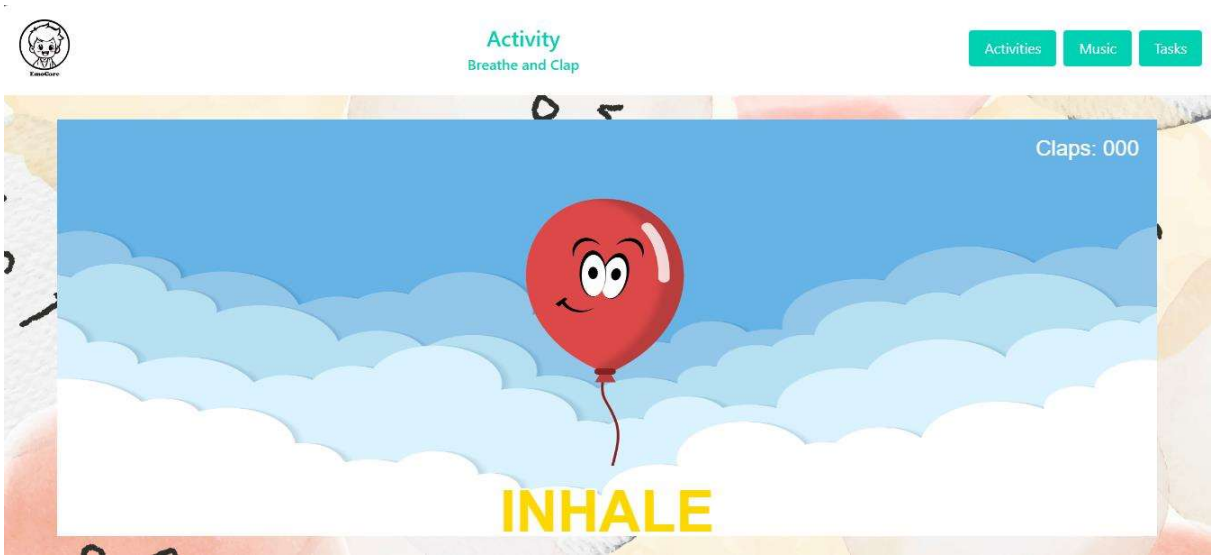

Figure 15 UI - Breathe and Clap Activity

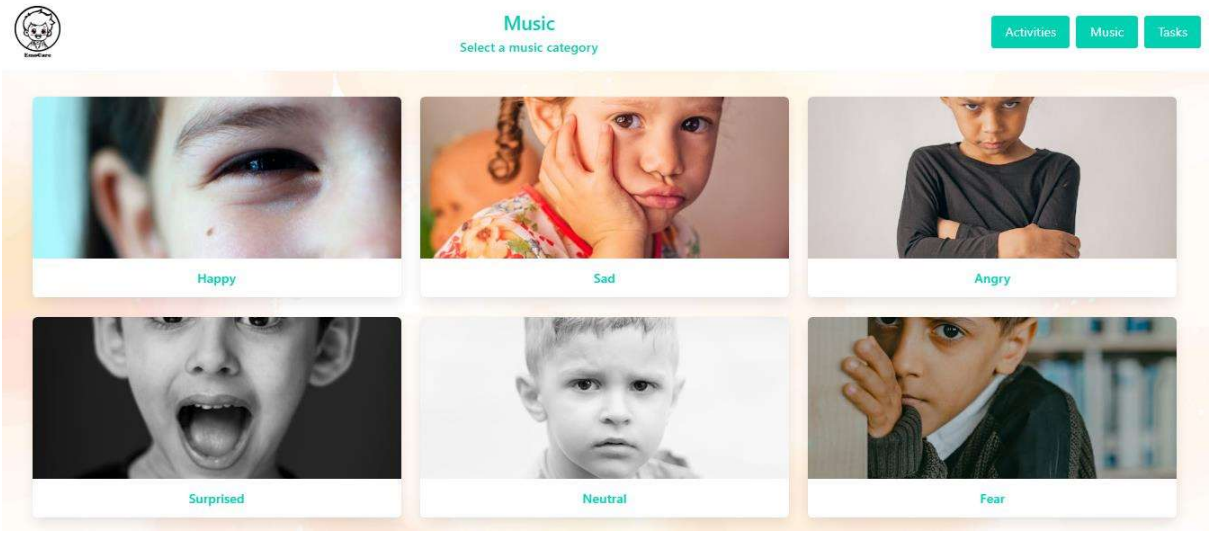

Figure 16 UI - Music Page

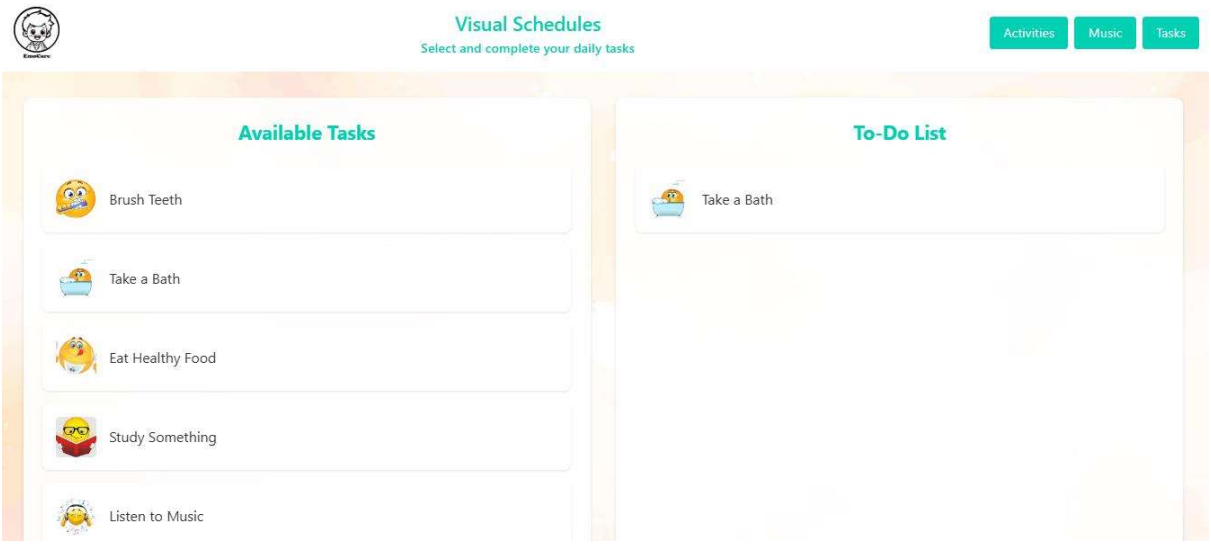

Figure 17 UI - Visual Scheduler Page

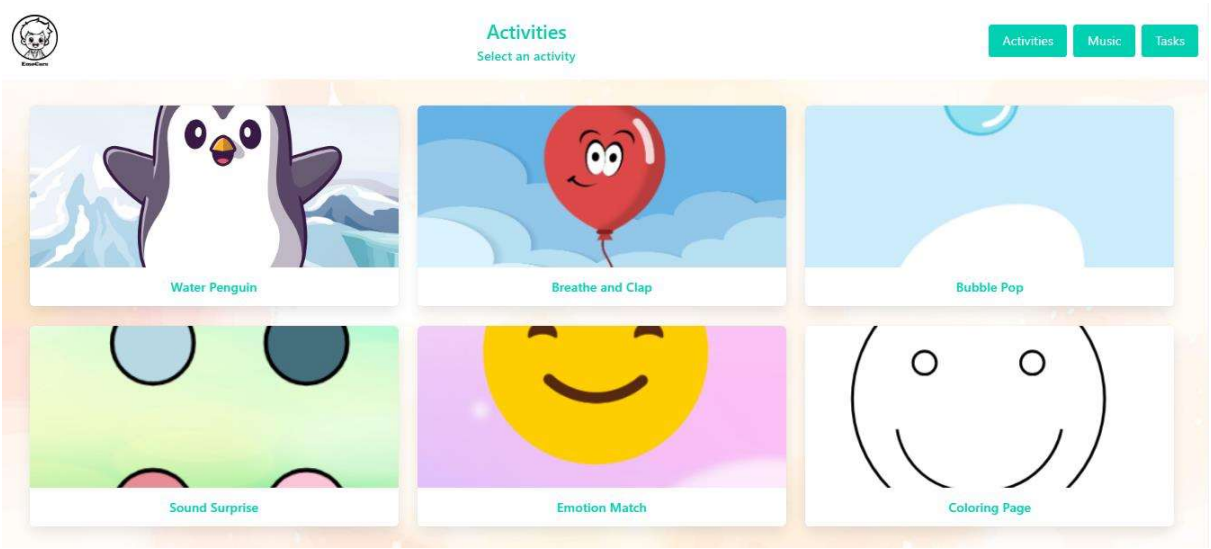

Figure 18 UI - Activity Page

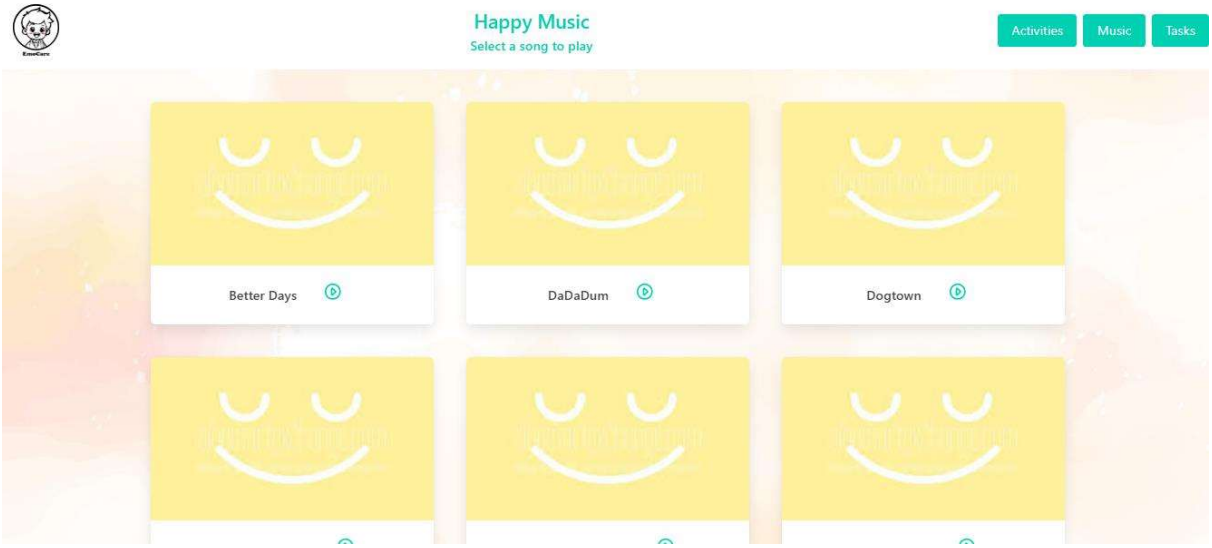

Figure 19 UI - Happy Music Page

15. USER MANUAL

EmoCare aims to develop emotional well-being in children with autism by recognizing emotions and offering interesting activities based on their feelings. The following guide will assist you in understanding the features of the website, from automatic recognition of emotions to interactive activities, music, and visual schedules. Read the instructions carefully to get the best out of EmoCare and to have a smooth experience.

15.1 HOME PAGE

On opening the EmoCare website, the app then begins recording video clips as soon as a face is detected. The user must be in proper lighted conditions for accurate detection.

- **Duration**: The detection process lasts for 10 minutes.
- **Emotion Display**: After detection, a modal appears displaying the detected emotion.
- **Correction Option**: If the detected emotion is incorrect, the user can manually select another emotion.
- **Redetection**: If desired, the user can choose to detect the emotion again.
- **Activity Redirection**: Once an emotion is confirmed, the app redirects the user to a predefined activity.

15.2 ACTIVITY PAGE

The Activity Page contains a collection of engaging activities for children.

- **Activity Selection**: Users can browse the full list and choose any activity.
- **Play Mode**: Upon selecting an activity, it starts immediately.
- **Interactive Features**: Activities are designed to be engaging and encourage emotional well-being.

15.3 MUSIC PAGE

Music can influence emotions, and EmoCare provides a dedicated section for emotion-based music.

- **Emotion-Based Playlists**: Each emotion has a corresponding playlist.
- **Music Selection**: Users can navigate to the music page and choose any playlist.

- **Audio Control**: Play, pause, options are available for controlling the music.

15.4 VISUAL SCHEDULE PAGE

A structured schedule helps children manage their daily tasks effectively.

- **To-Do List Creation**: Users can select items from a predefined list to create their personalized visual schedule.
- **Task Completion**: Tasks can be marked as completed by simply clicking on them.
- **Daily Planning**: This feature helps in organizing daily activities in a structured manner.

15.5 MENU BAR NAVIGATION

EmoCare provides easy navigation options through the menu bar.

- **Activities**: Directs to the Activity Page for selecting and playing activities.
- **Music**: Opens the Music Page for emotion-based playlists.
- **Tasks**: Access the Visual Schedule for managing daily routines.