Diverse Makers

Final Report

Version 1.0

12-13-2024

Project Sponsor: Dr. Jared Duval

Faculty Member: Isaac Shaffer

Project Mentor: Vahid Nikoonejad Fard

Team members:
Daniel Minichetti (Team Lead)

Kane Davidson

Eduardo De La Rosa

Elleana Negrelli

Aaron Ramirez

Overview

In this document, we provide a comprehensive examination of the Diverse Makers application's development, from its initial conception as a solution for improving STEM accessibility, through its technical implementation and testing, to recommendations for future enhancements, serving as both a final project report and a guide for future development teams.

Table of Contents

1.0 INTRODUCTION

Diversity in science and technology is vital for pushing boundaries and driving innovation within fields. By leveraging different backgrounds, experiences, and perspectives, STEM fields become enriched with unique insights that increase problem solving and breakthroughs. However, a significant disparity exists in STEM representation, while over 40 million Americans have a disability, that only 3% of the STEM workforce includes individuals with disabilities. Through critical research, we've identified this severe underrepresentation stems from a lack of hands-on, individualized learning environments that people with disabilities often learn best in. Traditional educational settings struggle to accommodate these needs due to large class sizes and conventional teaching methods in today's education systems.

Makerspaces offer a potential solution, as learning environments that provide hands-on, interest-based maker projects, and differentiated STEM education. However, many makerspaces lack the necessary accessibility features and learning resources to fully support and adapt to individuals with various disabilities. Additionally, many makerspaces lack the proper physical accessibility and assistive technologies, making it difficult for individuals with disabilities to effectively utilize these spaces. Current STEM materials and maker projects also pose challenges as these resources rarely accommodate diverse learning needs or disabilities, limiting their usefulness for potential users. Social isolation and a lack of community in STEM fields can also discourage talented individuals with disabilities from pursuing their STEM interests, perpetuating this cycle of underrepresentation.

Our client, Dr. Jared Duval is an assistant professor at the School of Informatics, Computing, and Cyber Systems at Northern Arizona University. He is also the director of the Playful Health Technology Lab, an interdisciplinary research team whose mission is to design, implement, and study human-computer interaction technologies that enhance the experience of improving and maintaining one's health. Utilizing research through design, Dr. Duval develops therapy games and playful applications that promote a better wellbeing, specializing in serious games for health that emphasize human-computer interaction with assistive technology.

To address this crucial gap in STEM education, Dr. Duval, envisions a solution that aims to increase awareness and opportunities for those with disabilities through a crowdsourced application designed to connect makerspaces to people of all disability backgrounds. In collaboration with Dr. Duval, we can empower those with disabilities to explore their passions and aspirations in STEM by creating a collaborative and supportive environment built around makerspaces that prioritizes accessibility at every step.

Our solution, Diverse Makers, is a cross-platform mobile application that serves as a centralized hub for sharing STEM resources, projects, and activities specifically designed for individuals with disabilities. The application transforms how people with disabilities engage with STEM education, connecting users to accessible resources and local makerspaces near them. Creating a network to support those pursuing their STEM education, our application provides several key features: An accessible interface that caters to multiple disabilities, a platform for makerspaces to upload STEM content, makerspace discovery to foster community, and user profiles with customizable accessibility settings for various needs. By bridging the gap between makerspaces and individuals with disabilities, Diverse Makers aims to create a more inclusive STEM ecosystem that not only provides access to valuable learning resources but enables a supportive community that can help break down social barriers in STEM fields.

The impact of this project extends beyond individual users as an accessible STEM education can create pathways for people with disabilities to interact with makerspaces. Diverse Makers makes a significant contribution by increasing diversity in STEM fields, creating an inclusive educational environment for all learners, and building a supportive community with makerspaces to better serve diverse populations. In this document, we will delve into the development process, technical implementation, and future potential of our application. In each subsequent section, we will explore the requirements, architectural design, and implementation of our app, describing each major module within our system. Additionally, we will outline our testing plan and project timeline to explain how our solution was realized and ready for use within the makerspace community. With these goals in mind, we hope to transform the makerspace landscape and bring a crucial tool that everyone can use to better their STEM education learning outcomes and pursue their passions.

2.0 PROCESS OVERVIEW

Our team followed an agile development process with a strong emphasis on accessibility and user-centered design. This approach enabled us to maintain continuous feedback loops with our client and key stakeholders, particularly individuals with disabilities, ensuring our solution effectively addressed real users' needs throughout the development cycle.

2.1 Team Organization and Roles

Each team member adopted specific responsibilities while contributing to the overall development of our application as a programmer:

- **Daniel Minichetti (Team Lead)**
    - Coordinated team assignments and tasks across each major process of the project to ensure meaningful progress is being made.
    - Set up regular meetings with our key stakeholders and client for each phase of the application development.
    - Provided support and led efforts to resolve conflicts within the team.

- **Kane Davidson (Architect)**
    - Guided major architectural decisions and ensured they were being followed throughout the development process.
    - Created the major development roadmap for key milestones in the development for the application.
    - Managed the high-level technical implementation of the application at each step of the development cycle.

- **Eduardo De La Rosa (Quality Assurance)**
    - Managed the code review process to ensure the code base was accurate and readable for each pull request committed to the project repository.
    - Coordinated the version control and branching to meet the high-quality standards throughout the major development processes.

    - Led testing initiatives to ensure the application was as accessible as possible and met the needs of all stakeholders.

- **Elleana Negrelli (Recorder)**
    - Documented meeting outcomes from crucial meetings with the team, mentor, and client for each critical phase of the project.
    - Managed project documentation for the application throughout its development.
    - Supported accessibility testing throughout the implementation of the application.

- **Aaron Ramirez (Customer Communicator)**
    - Communicated directly with our client as the primary point of contact for the team throughout each major phase of the project.
    - Gather key user requirements from our client to ensure the application covered the necessary accessibility features and functionality for the application's user base.
    - Coordinated user testing to validate the functionality and accessibility of the application for further refinement.

2.2 Development Lifecycle

Our development lifecycle followed several key phases to successfully execute the application’s key requirements and accessibility features:

2.2.1 Research Phase

- Conducted extensive stakeholder interview with our client
- Researched accessibility standards and guidelines
- Documented functional and non-functional requirements
- Created initial user stories and acceptance criteria

2.2.2 Design Phase

- Developed architectural design focusing on accessibility and cross-platform compatibility
- Created UI prototypes incorporating WCAG 2.1 guidelines and Nielsen's Usability Heuristics

2.2.3 Implementation Phase

- Adopted iterative development cycles
- Implemented core features in priority order:
    1. User authentication and profile management
    2. Accessibility settings and customization
    3. STEM resource hosting and management
    4. Makerspace discovery

2.2.4 Feedback and Refinement Phase

- Conducted comprehensive testing for our application:
    - Unit testing of core components
    - Integration testing of system interactions
    - Usability testing with disabled users
- Gathered and incorporated user feedback
- Refined features based on testing results

2.3 Development Tools and Practices

Our development process utilized several key tools and practices to ensure our team remained on-track for each deliverable and focused on the application's functionality:

2.3.1 Version Control

- **GitHub:** Code management and collaboration
    - Feature branches for new development and implementation
    - Pull request reviews for quality control and requirement satisfaction
    - Detailed commit messages documenting changes by each team member

2.3.2 Project Management

- **Google Calendar:** Scheduling and coordinating availability
    - Weekly team meetings (Fridays 1:00-2:00 PM)
    - Regular mentor meetings (Wednesday 5:30-6:00 PM)

- **Discord:** Team communication
    - Dedicated channels for different aspects of the project
    - Regular status updates and discussions
    - File sharing for quick collaboration
- **Google Workspace:** Documentation creation
    - Collaborative document editing for project assignments
    - Version control for documentation

2.3.3 Development Environment

- **Visual Studio Code:** IDE for application development
    - Extensions for React Native development
    - Live Share feature for collaboration
    - Plugins for automated testing and documentation
- **React Native:** Cross-platform development
    - Material Design components
    - React Paper for accessibility focused UI components
- **Firebase:** Backend services
    - Authentication
    - Real-time database
- **Jest:** Automated testing
    - Unit testing framework
    - Integration test suite

This structured process enabled our team to maintain high quality standards while ensuring our application remained focused on its mission to improve STEM accessibility. We implemented continuous integration with regular code reviews and testing to address any issues that occurred throughout each step of the development process, allowing us to fix potential bottlenecks quickly. We also held a strong focus on the application's accessibility throughout development, regularly consulting the Web Content Accessibility Guidelines (WCAG) and Nielsen's Usability Heuristics for accessibility compliance. With these processes we were able to not only refine but improve the application's functionality throughout the development cycle.

3.0 REQUIREMENTS

Our requirement acquisition process involved extensive meetings with our client, and in-depth research into accessibility standards. This process helped us identify and prioritize features that would make STEM education more accessible through makerspaces. With this in mind we will delve into the functional, performance, and environmental requirements needed for the development of our project, as required by our client. The functional requirements include the outcomes we completed to have a successful application that meets the needs of our users. Moreover, the performance requirements were objectives added to enhance the overall usage and aesthetics within the app. Lastly, the environmental requirements covered the restrictions placed on development plans or team members by software constraints that were outside our control.

3.1 Requirements Overview

Since our application will cater to a large group of users who wish to progress their STEM education, providing STEM learning in the most accessible way possible to individuals with a variety of disabilities was our top priority. Taking an in-depth look into our project, the following table covers the key requirements that were integrated into our application:

User Level Requirements	Functional Requirements	Performance Requirements	Environmental Constraints
1. Enable user profile creation and management 2. Provide accessible STEM resources 3. Connect users to local makerspaces 4. Facilitate communication between users and makerspaces 5. Enable STEM content sharing for makerspaces	1. Secure user authentication 2. Resource hosting for STEM learning content 3. Accessible user interface for multiple disabilities 4. Local makerspace discovery 5. Search and filter functionality for STEM content	1. Cross-platform compatibility for users on either IOS or Android devices 2. Optimized data retrieval for quick access to content 3. Responsive app layout for different devices and orientations 4. Real-time updates across application screens	1. Backend development with Google's Firebase technology 2. Compliance with modern accessibility standards 3. Adherence to user privacy regulations and app data

3.2 Functional Requirements

The functional requirements outlined below form the core functionalities of our application, ensuring it serves as an effective platform for connecting disabled users to not just STEM resources but makerspaces as well. The following subsections detail these key components and how they were implemented within our final system:

3.2.1 Secure User Authentication

The implemented authentication system provides the following functionality within our application:

- **Email and password-based authentication**
    - Users can create an accountant login using their email address and password
- **Secure Session Management**
    - Protects sensitive data and allows users to access the application securely
    - Implements secure token-based authentication

3.2.2 Resource Hosting for STEM Content

Since our application is a central hub for STEM resources, the implemented resource hosting system supports:

- **Multiple Content Formats**
    - Makerspaces can upload STEM resources in either a text, image, or video format
    - Easily accessible STEM content that caters to multiple disabilities and needs
- **Searchable Resource Database**
    - Searchable database stores all learning resources for easy access to activities
    - Efficient indexing for easy access to activities
- **STEM Activity Tagging**
    - Activities can be tagged and categorized for better searchability

3.2.3 Accessible User Interface for Multiple Disabilities

Our user interface implementation involved utilizing Nielsen's Usability Heuristics and to Material Design to provide an accessible user interface that enhances usability for all our users through:

- **Adjustable Font Sizes**
    - Font size can be adjusted for those with different vision needs
- **High Contrast Mode**
    - High contrast can be enabled for low vision user
- **Navigation Support**
    - Consistent layout patterns across screens for intuitive content interaction
    - Clear navigation support for those with cognitive impairments
- **Input Optimization**
    - Touch targets are optimized for those with motor impairments

3.2.4 Makerspace Connection

To allow users to easily connect with makerspaces, our makerspace discovery and connection system includes:

- **Makerspace Discovery**
    - Users can find any makerspaces nearby to enhance sense of community
- **Profile Integration**
    - Makerspace profile includes accessibility information and specialization
    - User can view makerspace contact information and address details
    - Connectivity improves makerspace resources hosted on application

3.2.5 Profile Creation and Management

To enhance user engagement and personalization, our profile creation and management system allows for:

- **Customizable User Profiles**
    - User can set a personal username and picture to share more about themselves
    - Disability tag can be set on profile to inform and connect with others
- **Activity History**
    - Activity history allows users to view a list of previously created activities
    - Populates profile page with created activities others can interact with

3.2.6 Search and Filter System

Given the amount of STEM resources available on the application, our search and filter system implements:

- **Advanced Search Options**
    - Keyword-based Search allows users to type specific keywords to find activities
    - Filtering options enables user to find activities with a specific tag
- **Search Result Management**
    - Sorts activities based on relevance for a user to easily find an activity
    - Search history tracking can display recent searches a user has made
    - Users can clear search results to erase previous search queries

3.3 Performance Requirements

Now that we have analyzed what our system accomplishes given our functional requirements, we will look at how those requirements are expected to perform under specific criteria. These non-functional requirements ensured our application was easy for disabled individuals to use and engage with. By prioritizing **usability, accessibility, optimization** & **maintainability** our system provides optimal system performance and user experience:

3.3.1 Usability

The application provides an intuitive experience not only for individuals with disabilities, but also for makerspaces hosting their resources on our platform through:

- **Interface Design**
    - Clear and consistent navigation patterns enable makerspaces to upload content onto our platform hassle-free
    - Intuitive content organization allows user to view resources in a clear format
    - Minimal learning curve allows new users to quickly access relevant STEM learning resources, connect with local makerspaces, and easily manage their account
- **Virtual Setting for Learning**

- Provides STEM learning opportunities to those unable to physically access local makerspaces around them
- Full application functionalities are available from any environment with an internet connection

3.3.2 Optimization

With a heavy focus on data retrieval, our application quickly provides users access to any learning resources. By taking advantage of Google Firebase, our system performance is maintained by:

- **Data Management**
  - Efficient data retrieval and caching allows optimal database performance
  - Optimized content delivery allows users to easily access any learning resource
  - Background synchronization provides a reliable experience across pages
- **Cross-Platform Support**
  - Consistent performance across iOS and Android operating systems
  - Responsive design for viewing content on different screen sizes

3.3.3 Maintainability

Maintainability was an essential non-functional requirement for updating our application over time. As our project develops, long-term sustainability and a reduction in complexity for developers is ensured through:

- **Coding Quality**
  - Modular architecture creates a solid foundation for potential future features as all the functional aspects of our source code can be modified while avoiding complex logical solutions
  - Consistent coding standards allows our source code to be readable with a clear understanding of what each individual component is responsible for within the system
  - Proper documentation adds beneficial context to our systems by informing developers the intended purpose of certain functionality for each feature in our application

- **Version Control**
    - By creating a Github repository our source code can be safely maintained and avoid any conflicting issues that may arise during development
    - A structured Git workflow provides a reliable method to document where potential bugs or issues occurred during our version history
    - Implementing a feature branching strategy enables new features to be added in a controlled environment without overwriting older versions of our application

3.4 Environmental Requirements

Having reviewed the necessary capabilities of our system, and how these components operate to meet our desired criteria, we'll delve into the environmental constraints that encompass the project. Our system considers many software and hardware requirements to operate within the following technical constraints:

3.4.1 Platform Requirements

Our application is built on two main structural components that work together to create a seamless and accessible experience for our user base:

- **Backend Infrastructure**
    - Google Firebase served as our backend framework providing real-time database functionality for instant data synchronization
    - Firebase authentication services provide secure user and session management
    - Firebase SDK integration enables scalable data management for STEM resources
    - Firestore enables structured data storage for efficient query handling and indexing
- **Frontend Infrastructure**
    - Utilizing React Native as our frontend framework provided cross-platform development capabilities for IOS and Android ensuring a consistent user experience across devices
    - Native UI component for both mobile operating systems ensures optimal performance for our application

3.4.2 Compliance Standards

Our application adheres to strict compliance standards to ensure accessibility and user privacy are not just secured but maintained throughout the user experience:

- **Accessibility Guidelines**
    - WCAG 2.1 compliance allows the application to have an operable user interface and navigation with robust content and reliable interpretation of information
- **User Privacy and Data**
    - User data is secured and privacy is maintained throughout an application session

The requirements outlined above reflect our commitment to creating a user-centered platform that bridges the gap between makerspaces and potential STEM learners. By implementing functional requirements such as secure authentication, an accessible user interface, and STEM resource hosting, we've built a foundation that serves our target users' needs. We successfully implemented all core requirements, with extra focus spent on the user experience and accessibility features. With React Native for the frontend and Firebase for the backend we were effectively able to meet our functional needs and environmental constraints, making sure our application can evolve while maintaining its commitment to accessibility.

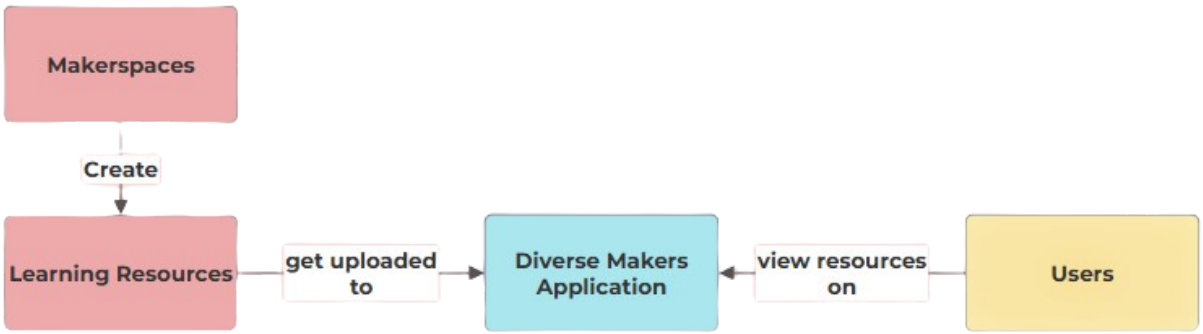
4.0 ARCHITECTURE AND IMPLEMENTATION

Our architecture implements a layered design that prioritizes accessibility, maintainability, and scalability. At its highest level, our system facilitates an interaction between makerspaces and users through our application. Makerspaces create and upload learning resources to the platform, which are then made available for users to view and engage with. This workflow enables the sharing of STEM educational content in an accessible format, creating a valuable connection between makerspaces and individuals with disabilities. The application serves to manage these interactions, handling everything from content upload and storage to user authentication and easy to use accessibility features. The following section will provide a detailed overview of our system's architecture as built, including component responsibilities,

communication patterns, and implementation decisions made throughout our development process. With these critical architectural components and implementations we can ensure both makerspaces and users can easily participate in this knowledge-sharing ecosystem while giving future developers a clear understanding of how to maintain and extend the system to keep its core mission.

4.1 Architecture Overview

The following diagram illustrates an overview of our architecture and flow demonstrating how makerspaces and users connect through our application:



4.1.1 System Stakeholders

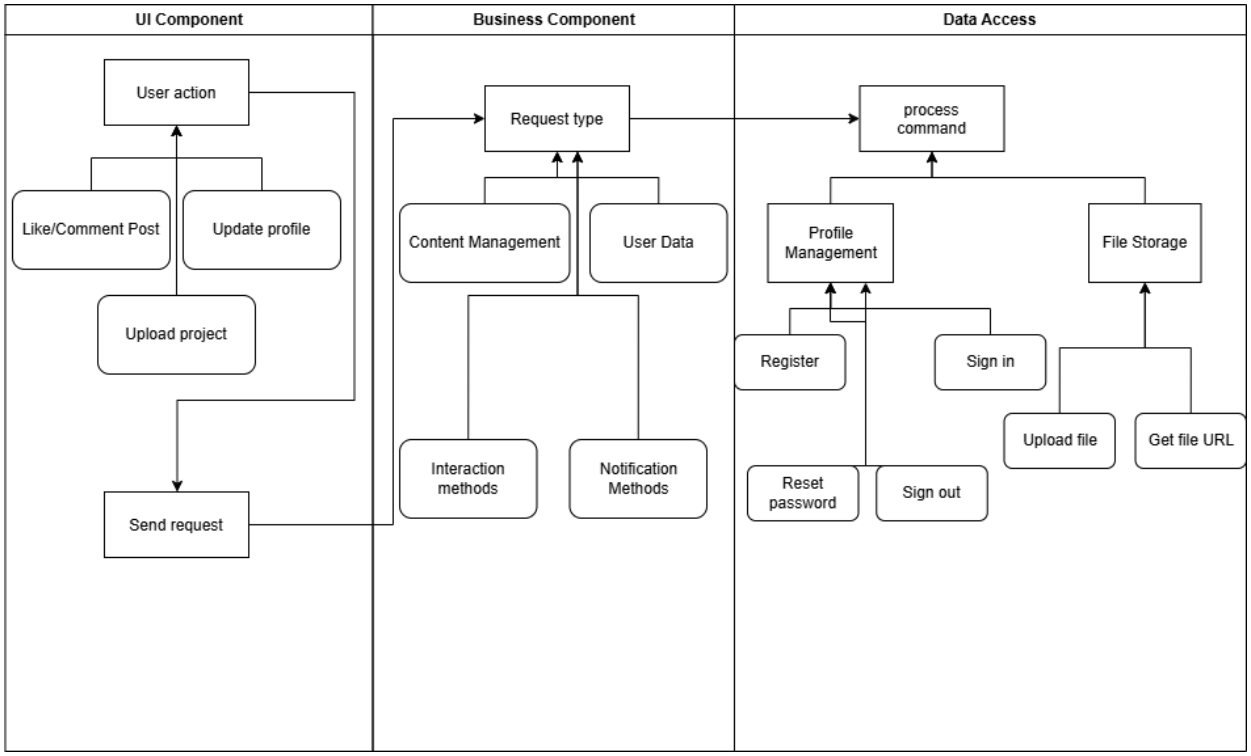
- **Makerspaces**
  - Represent physical maker communities and educational spaces
  - Create and manage STEM learning content
  - Upload resources through authenticated accounts
- **Learning Resources**
  - STEM educational content in multiple formats (text, video, and images)
  - Created and uploaded by makerspaces
  - Activities tagged for easy discovery
  - Optimized for accessibility needs
- **Diverse Makers Application**
  - Central platform managing all interactions
  - Handles user authentication and profiles

    - Processes and stores learning resources
    - Implements accessibility features
    - Manages real-time data synchronization
- **Users**
    - Individuals seeking STEM education resources
    - Access content through customizable interfaces
    - Can view and interact with learning materials
    - Connect with local makerspaces through the platform

This high-level architecture emphasizes our platform's role as a bridge between makerspaces and users, facilitating the sharing of accessible STEM educational content. Each component has been designed with accessibility and usability in mind so makerspaces and users can effectively participate in the STEM ecosystem.

4.2 Key Component Responsibilities and Features

When designing an application with accessibility in mind, having an intricate design plan was pivotal to providing the most user friendly experience possible. As we developed the key architecture for our application, it was important to define what purpose each component serves, and how it contributes to the overall design philosophy to be achieved. The diagram below highlights the three main components our application utilizes and their connection to one another:



4.2.1 User Interface Component

The user interface (UI) provides an intuitive experience that is accessible to all users of our application through customization of their interface such as changing font sizes, providing a high contrast mode, and other control over their experience.

- **Key Responsibility:** Provide an accessible, intuitive interface for all users
- **Features Implemented**
    - Responsive design for iOS and Android devices
    - Customizable accessibility (high contrast mode and adjustable font sizes)
    - User Profile Management
    - Content browsing and search functionality

4.2.2 Business Logic Component

The business logic component manages the core functionalities of our application. This includes processing user input, facilitating secure authentication, managing makerspace matching, and handling resource uploads. The business logic and data access components work

together to retrieve and store data, bridging the user interface to the backend and enabling seamless communication between the two for an optimal user experience.

- **Key Responsibility:** Handle core application functionality and data processing
- **Features implemented**
    - User Authentication and Authorization
    - Makerspace Matching based on user location, preferences, etc
    - Resource management (upload and tagging activities)

4.2.3 Data Access Component

The data access component was responsible for all interactions with the Firebase backend. It served as an abstraction layer between the business logic component and the database to manage data storage and retrieval operations. It has the following key responsibilities:

- **Authentication Management**
    - Handle user sign-up, login, logout, password reset, authentication state changes, and other related tasks using Firebase authentication
- **Database Operations**
    - Perform CRUD (Create, Read, Update, Delete) operations on user-generated content, profiles, and other data using Firebase Firestore
- **File Storage**
    - Manage uploading and retrieving media files (e.g., images, documents) with Firebase Storage
- **Real-time Updates**
    - Listen for and propagate real-time changes to keep the app content up to date
- **Security Enforcement**
    - Apply Firebase security rules to restrict unauthorized access and protect user data
- **Features Implemented**
    - Integration with Firebase real-time database and Cloud Firestore
    - Caching mechanisms
    - Data synchronization

4.3 Communication & Control Flow between Components

These components interact through well-defined interfaces that ensure data consistency and real-time updates across the application. Our communication architecture prioritizes reliable data flow while maintaining fast response time for accessibility features.

4.3.1 Client-Server Communication

Our application implemented a robust client-server communication model, leveraging Firebase's real-time capabilities and React Native's efficient state management:

- **Firebase SDK Integration**
    - React Native app communicates directly with the Firebase backend
    - Secure authentication token manage user sessions
- **RESTful API Implementation**
    - Structured endpoints for CRUD operations
    - Error handling and status codes
- **Real-time Data Synchronization**
    - Firebase real-time database enables instant updates

4.4 Component Use Case

To illustrate how these components work together, let's walk through typical use cases that demonstrate our system's core functionality and component interactions. These examples show how our layered architecture handles user interactions while maintaining accessibility:

4.4.1 Makerspace Creates STEM Activity

When a makerspace representative creates and uploads a new STEM learning resource, the following component interactions occur:

1. **UI Component Actions**
    - Displays activity creation interface
    - Provides tagging options for content categorization
    - Offers preview functionality before submission
2. **Business Logic Component Processing**

- Processes uploaded media for optimal storage
- Creates necessary database relationships
- Handles error cases and validation feedback

3. **Data Access Component Operations**
    - Stores activity data in Firestore
    - Uploads media files to Firebase Storage
    - Updates search indexes for discoverability
    - Creates real-time update triggers

4.4.2 User Discovers and Views STEM Activity

To demonstrate the full interaction cycle, here's how a user accesses the created STEM content:

1. **UI Component Actions**
    - Presents accessible search interface
    - Displays content with accessibility features
    - Handles user interaction tracking
2. **Business Logic Component Processing**
    - Processes search queries
    - Applies user preference filters
    - Manages view state
3. **Data Access Component Operations**
    - Retrieves activity data and media
    - Syncs user interaction data
    - Manages caching for performance

These use cases demonstrate how our components work together to create a seamless, accessible experience for makerspaces and users as we maintain system performance and data integrity throughout the application.

4.5 Architectural Style

Our application follows a layered architecture pattern that promotes separation of concerns and maintainability, enabling a scalable system that can serve our application's needs. This architectural approach organizes our application into different layers wither clear responsibilities and controlled interactions.

4.5.1 Layered Architecture pattern

The system implements a three-tier architecture where each layer has specific responsibilities and communication patterns:

- **Presentation Layer (View)**
    - React Native UI components
    - Accessibility feature implementation
    - User input handling and validation
    - Screen navigation and routing
- **Application Layer (Controller)**
    - Business logic implementation
    - Data transformation and validation
    - Session management
    - Error handling
- **Data Layer (Model)**
    - Firebase database operations
    - Data persistence and caching
    - Real-time synchronization
    - Security rule enforcement

4.5.2 Benefits of Layered Architecture

Several key benefits arise from our layered architecture approach:

- **Separation of Concerns**
    - Each layer has distinct responsibilities
    - Changes in one layer minimally impact others
    - Easier testing and debugging

- **Maintainability**
    - Modular component updates
    - Simplified dependency management
    - Easier feature additions and paths
- **Scalability**
    - Independent layer scaling
    - Optimized performance management
    - Flexible resource allocation

This architectural style proved to be effective in supporting our accessibility requirements making sure that our system was flexible and reliable. The clear separation between layers allowed us to evolve each component independently while ensuring consistent interaction patterns throughout the application.

4.6 Implementation Outcomes and Considerations

Our implemented architecture successfully delivered on our mission of connecting individuals with disabilities to STEM resources through an accessible, reliable platform. While our final implementation largely aligned with our initial architectural vision, one key change that was made during development to better serve our users' needs included:

- **User Interface Implementation**
    - Started with platform-specific UI designs
    - Consolidated to React Native for consistency
    - Improved maintenance efficiency while maintaining accessibility
- **Future Considerations**
    - Advanced social hub and collaboration features between users and makerspaces

The modular nature of our architecture ensures that these potential enhancements can be implemented without compromising the existing functionality or accessibility features that form the basis of our application.

5.0 TESTING

Software testing was vital in ensuring our application met its intended functionality while maintaining reliability and usability standards. This was crucial for our application as accessibility and ease of use are not just features but fundamental requirements. Software testing allowed us to systematically verify that each module functioned correctly in isolation, working properly when integrated with other modules, and providing a positive user experience. Through comprehensive testing, we not only identify issues but addressed them early in development, reducing inefficiencies while ensuring a high-quality final product. Additionally, through systematic testing, we verified that our implementation met both functional requirements and non-functional requirements. In the following section, we detail our testing methodologies, success criteria, and results for each testing type. This comprehensive testing plan helped ensure our Diverse Makers application was validated and successfully served its purpose of making STEM education more accessible to individuals with disabilities.

5.1 Testing Distribution Overview

Our testing plan encompassed three main testing stratgies, each targeting different aspects of the application and effort distribution:

1. **Unit testing** (50% of testing effort): Focused on validating individual components and functions, particularly around our core features
    - User authentication and profile management
    - STEM activity creation and block-based content management
    - Search and filtering functionality
    - Accessibility settings management

2. **Integration testing** (25% of testing effort): Verified the correct interaction between major system components
    - front end and back end communication via Firebase
    - Data flow between different application screens
    - Block manipulation and content updates

- Settings persistence across sessions

3. **Usability testing** (25% of testing effort): Ensured the application met our accessibility goals
    - Interface navigation and content discovery
    - Block-based content creation workflow
    - Accessibility features effectiveness (font size adjustment and high contrast mode)
    - Overall user experience for individuals with various disabilities

The structure and intensity of our testing plan was carefully designed to meet our application's unique requirements and user needs. This testing strategy is distributed based on component criticality and our user's needs. For example, the block-based content system receives intensive unit testing due to its central role in content creation and collaboration. Similarly, accessibility settings will undergo rigorous testing to ensure reliable functionality across different screen sizes and conditions. This approach ensures we allocate testing resources where they will have the greatest impact on user experience and application reliability.

5.2 Unit Testing

Unit testing is crucial due to our emphasis on accessibility and the need for reliable core functionality. With this in mind, we defined a unit as a distinct piece of functionality that can be tested independently. This includes individual methods, functions, and small components that serve specific purposes within our application. For example, a user authentication method, an activity manipulation function, or a settings management operation would each constitute a unit.

Given our focus on accessibility and increasing STEM learning outcomes, we allocated the largest portion (50%) to unit testing to ensure a solid foundation for our application's accessibility-related features and core functionality. For this phase, we used testing tools such as Jest to ensure that each component functions correctly on its own. After conducting several key test cases that checked for accessibility compliance and performance for responsiveness, we were able to obtain high test coverage across critical components with strong results in our accessibility features. Our unit tests achieved 95% coverage of core accessibility functions,

validated WCAG 2.1 compliance requirements, and confirmed proper functionality of features like font size adjustments and high contrast mode. These results gave us confidence in the reliability and accessibility of our individual components before moving on to integration testing.

5.3 Integration Testing

For our application, the integration testing phase was important due to our accessibility requirements. Even if individual components meet accessibility standards, the transitions and interactions between them must maintain accessibility without degradation. For example, a screen reader must seamlessly navigate between components, and font size changes must propagate consistently across all interface elements.

This phase accounted for 25% of our testing efforts and ensured that the components of our application work smoothly together, focusing on areas like data flow and consistent accessibility settings to ensure reliable data flow, state management, and smooth operation between our frontend and backend system. Our integration testing results demonstrated high system stability, with all key interfaces maintaining accessibility standards while achieving 100% data reliability in cross-component operations. Our results showed zero data loss during synchronization to the database and consistent accessibility settings across all interfaces. This gave us confidence that our application could provide a consistent, accessible experience across all user interactions.

5.4 Usability Testing

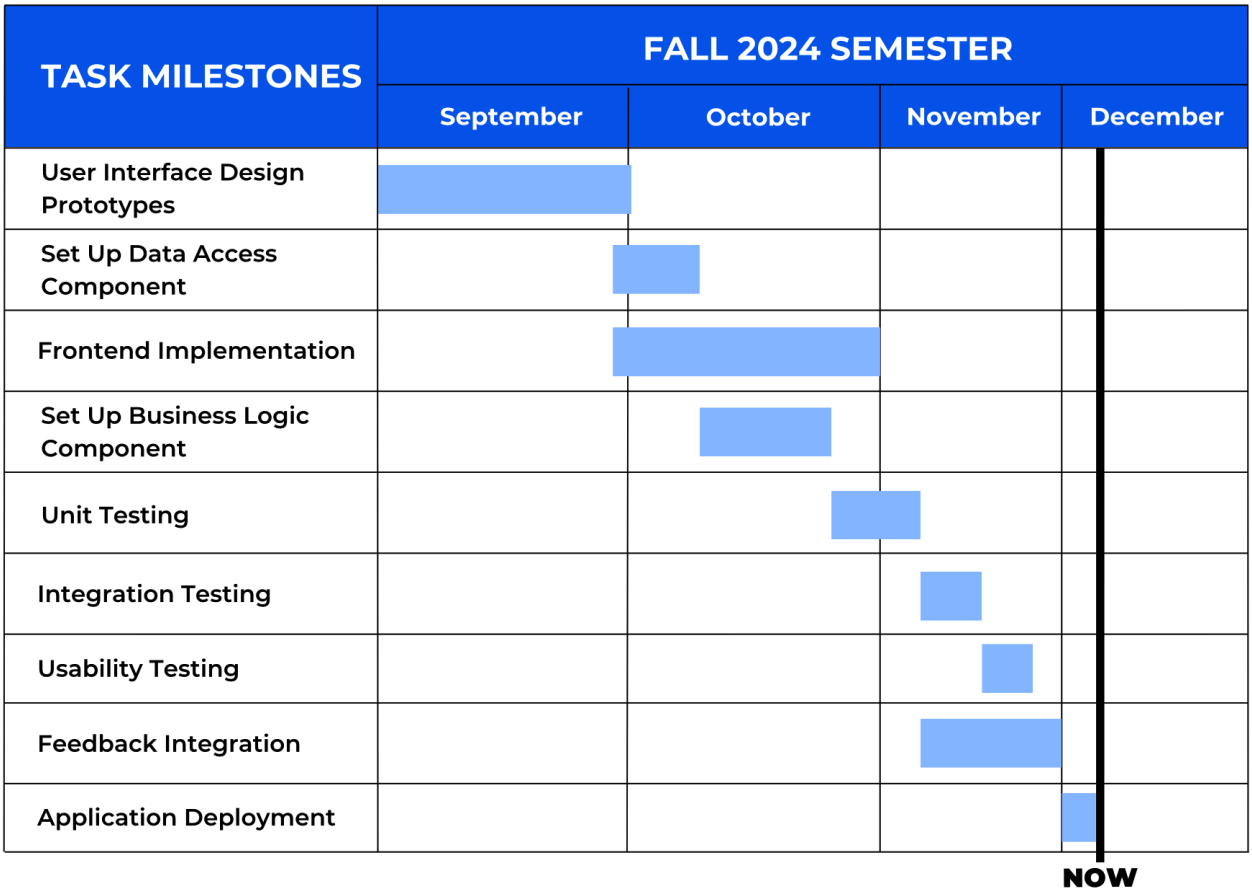
Usability testing is the critical evaluation method that focuses on how real users interact with our application, measuring their ability to accomplish intended tasks and their overall user experience completing workflows. For our application, usability testing was especially important as our primary users include individuals with multiple disabilities who required an accessible and intuitive interface for engaging with STEM learning resources.

Making up 25% of our testing efforts, we aimed to conduct structured testing sessions with diverse user groups to validate our application's accessibility and usability. Although we could not complete testing with the entire spectrum of disabilities we were still able to obtain key

feedback on the ease of navigation, content organization, and overall user experience from some disabled users. This user-centered testing approach helped us obtain positive feedback, allowing us to refine our final interface design to better serve our diverse user base while ensuring our STEM resources remain accessible to all.

6.0 PROJECT TIMELINE

During the fall semester our team focused on implementing the core application and conducting thorough testing. The gantt chart below illustrates the timeline of our project, tracking each phase of our application development throughout the entire semester for the different modules in our final system:



6.1 Development Phase

The following tasks were completed during the development phase which occurred from the start of September to the end of October:

- **User Interface Design Prototypes**
    - Created accessible interface designs supporting multiple disabilities
    - Implemented Material Design principles and Nielsen's Heuristics into our user interface design
    - Developed user profile and local makerspace discovery interfaces
- **Technical Implementation**
    - Set up our data access component with Firebase integration
    - Completed the frontend implementation in React Native with accessible UI components, cross-platform compatibility, and React Paper components integration
    - Established our business logic component including user authentication, content management system, and search/filter functionality

6.2 Testing Phase

The following tasks were completed during the testing phase which occurred from the end of October through the middle of November:

- **Unit Testing**
    - Core component functionalities, user authentication systems, content management features, and accessibility features
- **Integration Testing**
    - Frontend-backend communication, data flow validation, and cross-platform compatibility
- **Usability Testing**
    - End-user validation with disability groups, interface navigation testing, accessibility compliance verification, and user satisfaction assessment

6.3 Final Phase

The following tasks were completed during the final phase, which occurred from the start of November to the beginning of December:

- **Feedback Integration**
    - Incorporating user study results, implementing accessibility improvements, and refining user interface based on feedback
- **Application Deployment**
    - Final performance optimization, documentation completion, and platform deployment preparation

7.0 FUTURE WORK

While our current implementation successfully delivers on our initial goal of connecting individuals with disabilities to STEM resources, our development process and user feedback have identified several promising opportunities for future enhancement. These potential features would further strengthen the platform's ability to foster an inclusive STEM learning community.

7.1 Collaboration Features

The following collaborative features would enhance the social aspect of our application, furthering one’s connection to the STEM community around them:

- **Real-time Project Collaboration**
    - Enable multiple users to work on creating STEM activities together
    - Implement a shared workspace for group learning
    - Add collaborative note-taking and annotation tools for activities
- **Community Engagement Tools**
    - Create discussion forums for STEM topics
    - Enable peer mentoring connections
    - Implement achievement and progress sharing

7.2 Learning Features

- **Personalized Learning Paths**
    - STEM content recommendations
    - Customized difficulty progression
    - Progress tracking for activities

7.3 Accessibility Features

- **Enhanced Communication Support**
    - Sign language video integration
    - Speech-to-text capabilities
    - Multi-language support
- **Advanced Interface Customization**
    - Customizable gesture controls
    - Expanded keyboard navigation
    - Additional contrast modes

These proposed enhancements build upon our existing foundation while maintaining our commitment to accessibility and inclusion. The modular architecture we've implemented allows for these features to be added systematically, ensuring that each addition contributes to our mission of creating a more diverse and inclusive STEM community.

Our vision for Diverse Makers 2.0 extends beyond being just a resource-sharing platform to become a comprehensive ecosystem that nurtures STEM learning, collaboration, and community building for individuals with disabilities. Through these enhancements, we can further break down barriers in STEM education and create even more opportunities for diverse perspectives to enrich the field. The technical groundwork we've laid, combined with the positive response from our user community, positions Diverse Makers well for these future developments. We recommend prioritizing these enhancements based on user feedback and resource availability, with a focus on features that will have the most immediate impact on learning outcomes and community engagement.

8.0 CONCLUSION

Diverse Makers successfully demonstrates how technology can break down barriers and create new opportunities in STEM education. Through careful architecture, systematic development, and rigorous testing, we have created a mobile application that not only connects individuals with disabilities to makerspaces but fundamentally transforms how STEM resources are shared and accessed.

Our implementation achieves several key objectives: First, it provides a fully accessible platform that adapts to diverse user needs through features like adjustable font sizes and high contrast modes. Second, it creates a robust connection between makerspaces and learners, facilitating the sharing of STEM resources in formats that work for everyone. Third, it establishes a scalable foundation that can grow with the community's needs.

The technical architecture, built on React Native and Firebase, delivers a responsive and reliable experience across both iOS and Android platforms. Our layered approach to testing, from unit tests to usability studies, ensures that accessibility isn't just a feature but is considered into every aspect of the application. The positive feedback from user testing validates our approach and suggests real potential for impact in the STEM education landscape.

In collaboration with Dr. Jared Duval, Director of the Playful Health Technology Lab, we've created more than just an application, we've built a platform that empowers individuals with disabilities to pursue their STEM interests and connect with supportive communities. As this platform grows, it has the potential to help address the significant underrepresentation of individuals with disabilities in STEM fields by providing the tools, resources, and connections needed for success.

Diverse Makers represents a significant step toward a more inclusive STEM ecosystem. By bridging the gap between makerspaces and individuals with disabilities, we're not just improving access to STEM education, we're helping to build a future where everyone has the opportunity to contribute their unique perspectives and talents to the fields of science, technology, engineering, and mathematics.

9.0 GLOSSARY

Accessibility: The practice of making content and interfaces usable by people with various disabilities, including visual, auditory, physical, speech, cognitive, language, learning, and neurological disabilities.

CRUD Operations: Create, Read, Update, and Delete, the four basic operations of persistent storage in software applications.

Cross-Platform Development: The process of developing software applications that can run on multiple platforms or operating systems.

Firebase: Google Firebase is a set of cloud-based development tools that helps mobile app developers build, deploy and scale their apps.

Firebase Firestore: A NoSQL cloud database provided by Firebase for storing and syncing data in real-time.

High Contrast Mode: A display setting that increases the color contrast of the user interface to make text and interface elements more visible for users with visual impairments.

Jest: A JavaScript testing framework designed to ensure correctness of any JavaScript codebase.

Makerspaces: A place in which people with shared interests, especially in technology, can gather to work on projects while sharing ideas, equipment, and knowledge.

Material Design: A design system created by Google that provides guidelines for patterns, components, and tools supporting user interface design.

Nielsen's Heuristics: A set of user interface design principles developed by Jakob Nielsen for enhancing usability and user experience.

React Native: An open-source framework for building mobile applications using React and native platform capabilities.

React Paper: A collection of customizable and production-ready components for React Native, following Google's Material Design guidelines.

RESTful API: An architectural style for designing networked applications, emphasizing scalability and the use of standardized HTTP methods.

SDK (Software Development Kit): A set of tools for third-party developers to use in producing applications using a particular framework or platform.

WCAG: Web Content Accessibility Guidelines which are a set of recommendations for making web content more accessible to people with disabilities.

10.0 APPENDIX

10.1 Hardware

Our team developed the Diverse Makers application across multiple platforms to ensure cross-platform compatibility:

10.1.1 Development Platforms

Our team utilized multiple operating systems during development to ensure cross-platform compatibility and optimal performance across different environments:

- MacOS (Primary development platform)
- Windows 10/11

10.1.2 Hardware Specifications

Our team used a range of modern hardware configurations, from Apple Silicon to Intel processors. These specifications represent the hardware configurations that provided smooth development experiences, though lower specifications may work with some performance trade-offs:

- **Processor:** Apple M1/M2/M3 or Intel Core i5/i7/i9
- **Memory:** Minimum 4GB RAM (8GB recommended)
- **Storage:** 256GB SSD (minimum 15GB free space)

10.1.3 Minimum Requirements

While the application can be developed on lower-spec machines, we recommend the following minimum requirements be followed:

- Multi-core processor (2.0GHz+)
- 4GB RAM minimum
- 15GB available storage
- Stable internet connection
- Mobile device or emulator for application testing

10.2 Toolchain

Our workflow relied on several key tools and technologies for successful development and deployment of our application:

10.2.1 Core Development Tools

Our development workflow was built around a carefully selected set of core tools that provided the foundation for efficient development, testing, and deployment of our React Native application. These tools were chosen for their reliability, extensive community support, and specific features that enhanced our accessibility-focused development process:

- **Visual Studio Code**
    - Primary IDE for development
    - Essential for React Native development
    - Supports real-time collaboration
    - Key Extensions:
        - React Native Tools
        - Firebase Tools
        - Live Share
- **Node.js and npm**
    - JavaScript runtime environment
    - Package management

    - Version used: v22.9.0
    - Critical for running React Native
- **Expo CLI**
    - React Native development framework
    - Simplifies mobile app development
    - Handles build processes
    - Provides development server

10.2.2 Backend Services

Firebase served as our comprehensive backend solution, providing a scalable infrastructure that handled our application's data management, authentication, and storage needs. This tool was chosen for its real-time capabilities and extensive documentation:

- **Firebase**
    - Cloud-based backend service
    - Handles authentication
    - Real-time database
    - File storage

10.2.3 Additional Tools

To support our primary development tools, we utilized several supplementary tools that enhanced our development workflow, enabled effective collaboration, and ensured code quality through automated testing and continuous integration:

- **Git and GitHub**
    - Version control
    - Collaboration
    - Code review
    - CI/CD pipeline
- **Jest**
    - Testing framework
    - Unit testing
    - Integration testing

10.3 Setup

Follow these steps to set up your development environment to start developing the project and configuring the database:

10.3.1 Environment Setup

Setting up the development environment requires installing several key tools that work together to create our development ecosystem. Each tool serves a specific purpose in our React Native development workflow, from code editing to package management. Following these steps in order ensures all dependencies are properly configured:

1. Install Visual Studio Code
    a. Download from the official website: https://code.visualstudio.com/
2. Install Node.js
    a. On MacOS, run the following command: **brew install node**
    b. On Windows, download from the Node.js website: https://nodejs.org/en
3. Install Expo CLI
    a. Run the following command: **npm install --global expo-cli**

10.3.2 Project Setup

Once the core environment is configured, these steps will get you started with the actual project codebase. This process includes downloading the source code, setting up project-specific dependencies, and ensuring all necessary configurations are in place for local development. The commands should be executed in sequence to properly initialize your local development environment:

1. Clone the Github Repository
    a. Run the following command into your desired file location for the root directory: **git clone https://github.com/aaronr7734/Diverse-Makers**
2. Navigate to the Project Directory
    a. Run the following command: **cd DiverseMakersApp**
3. Install the Project Dependencies
    a. Run the following command: **npm install**

10.3.3 Firebase Configuration

The application requires the Firebase configuration to function properly, follow the next steps to get the database configured:

1. Create a new .**env** file in the root of the DiverseMakersApp directory
2. Access the Firebase Console: https://console.firebase.google.com/
3. Create a new project or select your existing project
4. Navigate to Project Settings > General
5. Under "Your apps", find your Firebase configuration object

Now, add the following variables to your **.env** file:

- FIREBASE_API_KEY=your_api_key
- FIREBASE_AUTH_DOMAIN=your_auth_domain
- FIREBASE_PROJECT_ID=your_project_id
- FIREBASE_STORAGE_BUCKET=your_storage_bucket
- FIREBASE_MESSAGING_SENDER_ID=your_messaging_sender_id
- FIREBASE_APP_ID=your_app_id
- FIREBASE_MEASUREMENT_ID=your_measurement_id

Replace each value with the corresponding value from your Firebase configuration. These credentials are essential for the app to communicate with Firebase services so keep them in a secure place. The **.env** file should never be committed to version control as it contains sensitive information.

10.3.4 Mobile Development Setup

Testing on real devices is crucial for ensuring proper accessibility feature implementation and user experience. The Expo Go application provides a convenient way to test the application during development without requiring a full native build process:

1. Install Expo Go on Mobile Device
    a. **IOS:** Download from the Apple App Store
    b. **Android:** Download from the Google Play Store
2. Ensure mobile device is on same network as development machine

10.4 Production Cycle

The production cycle for Diverse Makers follows a streamlined process that emphasizes code quality, accessibility testing, and efficient deployment:

10.4.1 Development Process

The development workflow centers around Expo's development server, which provides real-time updates and testing capabilities. This process allows developers to see changes instantly on connected devices or emulators, streamlining the development cycle:

1. Start Development Server
    a. Run the following command: **npx expo start**
2. Scan QR code for mobile testing
    a. Scanning code will bring you to the Expo Go application

10.4.2 Modifying Code Files

Code modifications follow a structured process to ensure changes are properly implemented and tested. The hot-reload feature in Expo also allows developers to see changes in real-time, making it efficient to implement and verify accessibility features and UI modifications:

1. Open File to Edit
2. Locate Function within File
3. Modify Code as Needed
4. Save Changes to the File
5. Changes Appear in Expo Go Application
    b. If edits are not immediately reflected the screen might need to be reloaded on the application for changes to take effect

10.4.3 Deployment Process

When ready to deployment to each

3. Build for Production
    a. For IOS, run the following command: **expo build:android**

b. For Android, run the following command: **expo build:ios**

10.5 Development Strategies

Our development workflow incorporates best practices for collaborative development, ensuring code quality and maintaining accessibility standards throughout the development process. These strategies have been refined throughout our development process to optimize team collaboration and code management while keeping our focus on accessibility requirements:

- **Adding New Features**
    - Create new branch
    - Implement feature
    - Create pull request
    - Review and merge
- **Updating Dependencies**
    - Check for updates: **npm outdated**
    - Update packages: **npm update**
    - Commit changes
- **Troubleshooting**
    - Clear cache: **npx expo start -c**
    - Check logs: **npx expo doctor**
    - Confirm Firebase configuration

This development environment and toolchain documentation provides the foundation for future development teams to continue building and improving the Diverse Makers application. By following these setup instructions and workflows, new team members can quickly establish their development environment and begin contributing to the project's mission of making STEM education more accessible. The success of future development efforts will be supported by this careful attention to environmental setup, allowing teams to focus on implementing new features and improvements rather than struggling with technical setup issues. We encourage future teams to maintain and update these instructions as new tools and better practices emerge.