Modern Principles of Software Development

SMART LEAP

1. Architecture First Approach

The architecture for the "Smart Leap" educational application revolves around leveraging mobile hardware, software components, and near-field communication (NFC) technology to facilitate interactive learning experiences for children aged three to six.

System's Structure:

1. Core Components:

- Mobile Device Hardware Utilization: Utilize components like the camera, voice recorder, and NFC capabilities for interactive learning experiences.
- Software Components: Developed a user-friendly Android-based application using Android Studio, emphasizing ease of navigation and interaction for kids.

2. Functional Modules:

- Learning Modules: Divided into two categories user module and admin module. They are subdivided into categories such as Alphabets, Numbers, Fruits, Vegetables, Animals, Colors, Drawing, and Quiz, each offering interactive lessons through pictures, voice guidance, and hands-on activities.
- Parental Access Control: Implement a monitoring system that allows parents to oversee and control their child's activities within the application, ensuring a safe learning environment.

Interactions Between Modules:

1. NFC Integration:

- Utilize NFC tags to enable seamless access to different learning modules. Each tag corresponds to a specific educational topic (e.g., Alphabet tag directs to Alphabet lessons).
- Interaction with Hardware: Implement functionalities such as drawing exercises using the device's touchscreen and torch for learning activities.

Development Phases:

Phase 1 - Initial Development:

- Focus on creating a robust Android-based application framework with functional modules for alphabets, numbers, and basic learning categories.
- Establish NFC integration and preliminary access control mechanisms.

Phase 2 - Enhancement and Expansion:

- Refine existing modules and introduce new interactive learning components for additional topics like animals, colors, and advanced exercises.
- Strengthen access control features for parents, allowing for better monitoring and customization of learning experiences.

Phase 3 - Optimization and Testing:

- Conduct thorough testing across different devices, ensuring compatibility and stability.
- Optimize performance, improve user interface elements, and address any identified bugs or issues.

Phase 4 - Release and Continuous Improvement:

- Launched the application, and gather user feedback to guide future updates and improvements.
- Prioritize continuous updates based on user input, technological advancements, and educational trends.

2. Iterative Life Cycle Process:

Planning:

- Define the overall project goals, scope, and target audience (children aged three to six).
- Identify the key features and learning modules (Alphabets, Numbers, Fruits, Vegetables, Animals, Colors, Drawing, Quiz) for initial development.

Design:

- Create a detailed architectural framework for the application, emphasizing the utilization of mobile hardware and software components.
- Design individual modules specifying their functionalities, interactions, and interfaces.
- Establish the flow of NFC-tag-based learning access and parental access control mechanisms.

Implementation:

- Develop the Android-based application using Android Studio, emphasizing the integration of hardware components like the camera and NFC for interactive learning.
- Create specific modules for each learning category (Alphabets, Numbers, etc.) with interactive features (voice, pictures, hands-on activities).

Testing:

- Conduct rigorous testing of each module for functionality, user-friendliness, and responsiveness across different devices.
- Perform NFC tag testing, ensuring seamless access to learning modules.
- Implement parental access control features and test their effectiveness in monitoring and controlling child activities.

Deployment:

- Release the application in phases, starting with core functionalities like Alphabets and Numbers.
- Gather user feedback on usability, engagement, and any identified issues or suggestions.

3. Component-Based Approach:

Identification of Components:

- Identify distinct educational topics/modules (Alphabet, Numbers, Fruits, Vegetables, Animals, Colors, Drawing, Quiz) as individual components.

Modular Development:

1. USER MODULE

   This module is an Android application that includes registration of the student, choosing a topic to learn, sending queries by parents, and sending feedback relevant to the topic. this enables the user (parents/teacher)to register into the system's database in order for the user to sign into the application.

- Registration

  Through this process the user registers in the app for kids.

- The Login

This enables the user to login into the application, using their registered details(username and password)stored in the database.

- Alphabets

  Through this children can learn the alphabet using pictures and voice.

- Numbers

  Through this children can learn numbers using pictures and voice.

- Fruits

  Through this children can learn about fruits using pictures and voice.

- Vegetables

  Through this children can learn and know about vegetables using pictures and voice.

- Animals

  Through this children can learn and know about animals using pictures and voice.

- Draw

  Through this children can learn to draw using colors

- Colors

  Through this children can learn about colors using pictures and voice.

  Quiz

  On learning all the related modules the child can attend quiz.

2. ADMIN
   This is a website developed using PHP. This module is server-side module for adding subjects, topics, and graphics related to the topic. Manage students and process feedback.

- login

  this is used by the administrator to enter the website.

- View users

This is used by the administrator to view the users registered in the app. This is a website that gets information about the users registered through the learning app.

- The Activator

The kiosk activator class is responsible for triggering the kiosk activity and stopping the user from leaving the application.

- The Virtual Teacher

This implements a kind of text to speech functionality in which the(parent)user can input words in which the (child)user is able to play back.

- The Drawing Torch Art

This allows the child to practice handwriting and other art practices, it implements android torch and draw in other for the child to use the torch screen and other functionalities.

Reusable Components:

- Design components with a focus on reusability across different parts of the application.
- Ensure that elements like voice prompts, interactive visuals, and educational content can be easily integrated and reused across multiple modules.

Integration and Cohesion:

- Integrate developed modules seamlessly into the application's framework, ensuring cohesive navigation and consistent user experience.
- Implement a cohesive design language and interactive features across all components to maintain continuity and ease of use for young learners.

Testing of Components:

- Conduct thorough testing of individual components to ensure they function independently and in harmony with the overall application.
- Verify the effectiveness of each module in achieving its educational goals and engaging the target audience.

Iterative Refinement:

- Gather feedback from users (parents, teachers, and children) after initial deployment.
- Refine and enhance individual components based on feedback, addressing any identified issues, improving user interactions, and expanding content if necessary.

By adopting a component-based approach, the "Smart Leap" application ensures that each educational module/component is developed independently, enabling easy integration and maintenance. This approach allows for scalability, reusability, and efficient management of the application's various educational elements.

4. Change Management System:

Maintenance

The maintenance phase focus on change that is associated with error correction, adoptions required as the software's environment evolves, and changes due to enhancement brought about by the user. Software will undoubtedly undergo change after it is delivered to the user. The "SMART LEAP" project uses Java and is object-oriented so that the user can have the concept of reusability over the software component to add or modify some modules. So maintenance becomes easier. When the system is in the maintenance phase some people within the system are responsible for collecting maintenance requests from users and other interested parties. System maintenance is the activity that occur following the delivery of the software product enhancement to software products adapting products to new environment and correcting errors.

- Corrective maintenance

Problem correction involves modification and revalidation of software to correct errors. The process that includes the diagnosis and correction of one more error is known as CORRECTIVE MAINTANCE.

- Adaptive maintenance

Adaptive maintenance results in the modification of the software to accommodate changes to its environment. As the external environments are changed in the future, the changes can also be accommodated.

- Perfective maintenance

Perfective maintenance extends the software beyond its original functional requirements. The required additional functions are easily added to the system. These additional functions enhance the system functionality and the system becomes more user-friendly.

- Preventive maintenance

Preventive maintenance must be conducted to enable the software to serve the needs of Its end-users. In essence, preventive maintenance makes changes to computer programs so that they can be more easily corrected, adapted, and enhanced.

5. Round Trip Engineering:

Round-trip engineering involves the ability to make changes to code and have those changes reflected in the system's higher-level designs. While my project doesn't explicitly mention this, the development process appears to allow for adjustments based on maintenance needs. By implementing Round Trip Engineering practices, the "Smart Leap" project can ensure that its design documentation stays synchronized with the codebase, facilitating better understanding, maintenance, and future development.

6. Model-Based Evolution:

Although not explicitly named, my project could adopt a model-based evolution approach by continuously refining and evolving the software based on user feedback, technological advancements, and changing educational needs.

7. Objective Quality Control:

System testing

System testing is a stage of implementation that aims the assure that the system works accurately and efficiently before live operation commences. System testing makes logical assumption mat all the parts of the system are correct, the goal will be successfully achieved. The testing phase is an important part of software development. It performs a critical role in quality assurance and ensuring the reliability of the software. It is the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied, The goal of testing is to cover requirements, design, or coding error programs. Consequently, different levels of testing are employed. The project “SMART LEAP” has done the system testing.

Unit testing

Unit testing focuses first on the modules in the proposed system to locate errors. This enables to detect errors in the coding and logic are contained within that module alone. Those resulting from the interaction between modules are initially avoided. In unit testing step each module has to be checked separately.

Integration testing

The objective of the integration testing is to take unit tested modules and to build program structure. Data can be lost across an interface. One module has adverse effect on another. Sub functions when combined do not produce the desired goal. Integration testing is done to create a program structure uncovering the errors across an interface all the modules are combined and tested as a whole.

Validation testing

After the termination of integration testing the solution is completely assembled as a package. A common problem with computer system is that it is very easy to put incorrect data into them. After the integration of the modules the validation testing for the system starts. The validation succeeds when the software function in a manner that can be responsively expected. After the validation test has been conducted one of the three possible conditions exists:

- The function or performance characteristics confirmed to the specifications are accepted.
- A deviation from specification is uncovered and a deficiency list is created.
- Proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

Output testing

After performing the validation testing, the next step is output testing of the proposed system since no system could be useful if it doesn’t produce the required output in the specific

format output obtained is tested by members of other groups and this testing does not result in any correction in the system.

Whitebox testing

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independent paths in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu.

8. Evolving Levels of Details:

Feasibility Study, System Analysis, Software Requirement, Module Description, System Specification, Design and Development Process, Testing and Implementation, and Maintenance are the phases in the development. Throughout each phase, this project gradually progresses from an overview of concepts and requirements to more detailed aspects. This progression aligns with evolving levels of detail, ensuring a structured and gradual approach to development without immediately delving into intricate technical or implementation specifics at the project's inception.

9. Establish a Configurable Approach:

The project demonstrates a well-defined and structured process from feasibility study to implementation and maintenance. This process can be adapted and configured for similar projects, indicating a configurable approach.

10. Demonstration Based Approach:

The emphasis on thorough testing, especially validation testing with varied inputs, aligns with a demonstration-based approach. This approach ensures that the system behaves as expected and delivers intended functionality, validating its usability and reliability.