SJSU SAN JOSÉ STATE UNIVERSITY

Project Workbook
On

CodeWizard – An Interactive Platform to learn Computer Programming

By

ANSHUL MITTAL
KARAN MANISH THAKKAR
NITISH POTDAR
NRUPESH PATEL

Advisor
PROF. RONALD MAK

December 9, 2017

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 LITERATURE SEARCH

Microsoft chairman, Bill Gates once said that: "Learning to write programs stretches your mind and helps you think better, creates a way of thinking about things that I think is helpful in all domains." Coding helps students to become more active and empower them with problem-solving and thinking skills not only in technical but also in other fields. Learning to code helps the students to broaden their intellectual mind-set and helps them to think in a creative way. Therefore, there is a need to provide the students with a platform where they can develop these skills and learn to code in a systematized way.

A flowchart is a structured way to represent processes or computer programs. Flowchart is used to design and document a complicated process into simpler small processes. These processes are identified by certain shapes and connected by one-sided arrows to depict the order of occurrence of each process. To design a flowchart, one need not have the knowledge of any programming language. It was first introduced in 1921 in a presentation to the members of American Society of Mechanical Engineers. The American National Standards Institute (ANSI) set standards for flowcharts and their symbols in the 1960s and the International Organizations for Standardization (ISO) adopted the ANSI symbols in 1970. The current standard was revised in 1985. The flowcharts generally flow from top to bottom and left to right. Flowcharts were introduced in computer science courses to teach algorithms and programming concepts to students. They are an effective way to write and study algorithms and programs. Flowcharts are an integral part of procedural programming, where the programmer uses it to articulate the programming logic. A complex program logic can be well modelled using a flowchart and the programmer can use this flowchart to explain the logic to its peers without the need of explaining the code. Thus, other programmers can use the program logic and convert it into any other programming language as needed.

Flowgorithm is an application which helps the user to create computer programs using simple flowcharts. It is a windows application which provides a blank canvas to the user and a palette of various shapes to choose from indicating various processes or actions. Upon selecting a shape, the user can input text related to the chosen shape in a pop-up textbox or the action to be taken by the program corresponding to the shape. The shapes are categorised into categories such as: input/output, declaration, control and looping. On completing the flowchart, the user when clicks on the generate button and the application generates the source code for the user in languages such as C#, C++, Java, JavaScript, Python.

One research paper states that the lack of problem-solving skills are exacerbated by complicated language syntax for novice programmers. It proposes a flowchart-based intelligent tutoring system which aims to generalize the early stages of learning programming. This system assists the programmer throughout the online tutoring materials and defines a correct dynamic learning sequence to understand a specific concept in programming. The system architecture consists of Bayesian Networks which offers personalised environments for users, a knowledge base which form the repository of lecturer notes, sample quizzes and questions along with their solutions, a user interface module where the users can interact with the system and adaptive guidance which offers the users with navigation support based on the knowledge level generated by the knowledge base module. (Hooshyar D. et al., 2015)

Another paper targets the recognition of handwritten flowcharts. It uses the dynamic programming to group the stroke sequence into stroke combinations. The recognition of a sketch diagram is divided into three steps: segmentation, classification and recognition. Segmentation step provides all the stroke combinations excluding the text. Then, these stroke combinations are labelled by the classification step. The labelled combinations, also known as candidate symbols, are compared with the actual symbols from the database and sorted based on the probability of similar occurrence to an annotated symbol.

The sequence of flowchart strokes is numbered and their spatial information is stored. Dynamic processing is used to classify stroke combinations into stroke groups. A limitation is set on the maximum number of strokes in a symbol. Thus, the identification of the candidate symbols becomes easier. This method achieves an accuracy of 91.4% if all the candidate symbols contain all the strokes. (Chen et al., 2015)

One paper discusses the traditional just-put-it-on-the-web educational system and proposes a new adaptive and intelligent web-based educational system (AIWBES). AIWBES attempt to be more interactive and builds a model of goals, preferences and knowledge level of each individual student using the system. Adaptive educational system use individual student account and progress information to provide a tailored and customised view of the system to each student. Intelligent educational system use artificial intelligence to provide a broad and better perspective of the system to students. The main goal of interactive web-based education system is to provide the students with intelligent help and adequate support to learn the topics and technologies in a sequential manner. Adaptive and intelligent web-based educational system provides a new development

platform which offers smart and personalised exposition of the resources available to the students. (Brusilovsky P. & Peylo C. 2003)

Recognition of handwritten text by computers has been a challenging problem since the advent of artificial intelligence. High-end handheld devices like mobile phones and tablets that come with a stylus can have a flawless handwriting recognition system, benefiting several users who are inclined towards taking down information by hand. The increasing technological advancements in the field of machine learning motivated several engineers to attempt develop various techniques for handwriting recognition. One of the papers discussed two fundamental approaches for handwriting recognition, viz. segmentation-based approach and segmentation free approach. Segmentation based approach involves character recognition, in which the words are divided into individual characters. Segmentation-free approach involves techniques which process the words without segmenting the text in the document. Segmentation free approach is further divided into two categories – learning based technique and example based technique. Although learning based approach provides a better performance than example based approach, it requires a high quantity and quality of dataset. (Kassis M. & El Sana J., 2016)

Samsung Galaxy high-end devices come with a stylus, S Pen, which helps the user to perform various tasks on the phone canvas. Using the S Pen, the user can input text in their own handwriting, which is recognised by the handwriting recognition software in real-time and converted into digital text. With the help of S Pen, one can also draw or edit objects using numerous types of pen strokes available on the canvas. S Pen allows the user to also accomplish all the activities that can be performed by human finger touch input. All these features have been stocked into the extremely revolutionary S Pen software development kit (S Pen SDK). The S Pen SDK also allows to develop applications which can harness the abilities of S Pen software for text and image recognition. S Pen SDK provides the verification ability if the S Pen is activated, identify the co-ordinates of the touch event, sensing the pressure of touch and verifying if the button is pressed while touching. The S Pen SDK helps the software developers to build an application which can utilize all these capabilities to provide different options in the application.

1.2 STATE OF THE ART

Some of the advanced tools and libraries that are open sourced for machine learning are described below –

1. Available SDK's for handwritten recognition

   WritePad SDK 5.2: WritePad is a natural, style, writer and lexicon independent multiplatform handwriting recognition technology. WritePad SDK enables natural handwriting input on any Android-powered device. It is currently available in 12 languages. WritePad SDK includes WritePad recognition engine in object code and dictionaries for all supported languages. It contains C/C++ header files with definition of API calls and structures and, lastly source code in C (JNI) and Java that demonstrates how to use the WritePad API. WritePad SDK supports Android 4.x through 5.x, including arm, arm-v7a, arm64-v8a, x86, x64, mips, and mips64 architectures. It supports handwritten gesture recognizer, several standard gestures, such as space, tab and return characters, cut, copy, paste, undo, etc. and a new shape recognition API which recognizes simple geometrical shapes, such as circle, rectangle, triangle, and straight line used in flow diagrams.

   The WritePad SDK key features include Recognition of natural handwritten text in a variety of handwriting styles: cursive (script), PRINT and MIXed. It recognizes dictionary words from its main or user-defined dictionary, as well as non-dictionary words, such as names, numbers and mixed alphanumeric combinations. Provides automatic segmentation of handwritten text into words and automatically differentiates between vocabulary and non-vocabulary words, and between words and arbitrary alphanumeric strings.

   MyScript: MyScript technology combines digital ink management with easy searching of handwritten text, as well as the accurate recognition of complex mathematical equations, geometric shapes and music notation. MyScript solutions are available for mostly all desktop and mobile operating systems.

   MyScript Diagram manages handwritten input of a wide range of diagrams, from orgcharts, to flowcharts, to mindmaps, etc. More than 10 types of shapes, and text in about 60 languages, and diverse types of connectors are supported. Editing and positioning gestures allow for powerful editing. Export to formats like SVG or GraphML makes the integration easy.

2. MongoDB: MongoDB is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with

schemas. Fields in a MongoDB document can be indexed with primary and secondary indices. MongoDB provides high availability with replica sets. MongoDB scales horizontally using sharding. MongoDB can be used as a file system with load balancing and data replication features over multiple machines for storing files. This feature can be used to store all the handwritten data to use for analytics and Machine learning purposes.

3. SQLite: SQLite is a relational database management system contained in a C programming library. SQLite is a popular choice as embedded database software for local/client storage in application software. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems such as mobile phones. SQLite has bindings to many programming languages.
4. Firebase: Firebase provides a real-time database and backend as a service.

   It provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. The service also provides client libraries that enable integration with Android, iOS, JavaScript and Node.js applications. The database is also accessible through a REST API and bindings for several JavaScript frameworks such as AngularJS, React, Ember.js and Backbone.js. Firebase has Storage services that provides secure file uploads and downloads for Firebase apps which can be used to store images, audio, video, or other user-generated content. Firebase has Hosting services that provides static and dynamic web hosting service. It supports hosting static files such as CSS, HTML, JavaScript and other files, as well as dynamic Node.js support through Cloud Functions. The service delivers files over a content delivery network (CDN) through HTTP Secure (HTTPS) and Secure Sockets Layer encryption (SSL).
5. Samsung S Pen SDK: The S Pen SDK provides libraries in an Android environment so that developers can develop various applications using S Pen. Pen SDK allows you to develop applications that use handwritten input. Pen SDK uses pens, fingers or other kinds of virtual pens for input. The S Pen SDK consists of two packages containing S Pen SDK-related Java Document and S Pen SDK Library. Pen SDK supports features like Advanced drawing brushes and canvas, Editing features for the MS-Word usage and Handwriting recognition for mathematical formula, shapes and 60 languages.
6. Existing compilers that can be integrated into android apps

Tesseract OCR: Tesseract is an OCR engine developed at Hewlett-Packard Laboratories with some more changes made in 1996 to port to Windows.

In 2005 Tesseract was open sourced by HP. Since 2006 it is developed by Google. The latest stable version is 3.05.01, released on June 1, 2017 and available on GitHub. This package contains an OCR engine - libtesseract and a command line program - tesseract. Tesseract has unicode (UTF-8) support, and can recognize more than 100 languages "out of the box". Tesseract supports various output formats: plain-text, hocr(html), pdf, tsv, invisible-text-only pdf.

7. TensorFlow: TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and used for machine learning applications such as neural networks. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open source license on November 9, 2015. TensorFlow is a library for numerical computation using data flow graphs. The graph nodes represent mathematical operations, while the graph edges represent the multidimensional data arrays that flow between them. This flexible architecture lets you deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code.

1.3 REFERENCES

Kassis M. & El-Sana J. (2016). Word spotting using radial descriptor graph. Paper presented at the *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR),* 31-35. doi:10.1109/ICFHR.2016.0019

Chen Q., Shi D., Feng G., Zhao X. and Luo B., "On-line handwritten flowchart recognition based on logical structure and graph grammar," *2015 5th International Conference on Information Science and Technology (ICIST)*, Changsha, 2015, pp. 424-429. doi: 10.1109/ICIST.2015.7289009

Brusilovsky P., Peylo C., “Adapative and IntelligentWeb-based Education Systems”, *International Journal of Artificial Intelligence in Education, IOS Press*, 2003, pp. 156-169. Available online at http://www.pitt.edu/~peterb/papers/AIWBES.pdf

Hooshyar D., Ahmad R. B., Fathi M., Yousefi M. & Hooshyar M., "Flowchart-based Bayesian Intelligent Tutoring System for computer programming," *2015 International Conference on Smart Sensors and Application (ICSSA)*, Kuala Lumpur, 2015, pp. 150-154. doi: 10.1109/ICSSA.2015.7322528

CHAPTER 2

PROJECT JUSTIFICATION

2.1 PROJECT JUSTIFICATION

Computational thinking is critical to foster programming aptitude amongst kids. The syntactical complexities of programming languages are better off not to be exposed to kids at their early stages of learning how to code, and rather having them learn how to develop an algorithm to solve a given problem. Flowcharts provide a pictorial view to represent the solution of a problem step by step using several shapes. An interactive platform where kids are first taught how to draw flowcharts using some examples and then giving them questions to practice would definitely garner their attention and help develop their logical aptitude.

In today's digital era, even with multifarious features integrated into handheld devices, writing with hand always seems to be more natural and intuitive. There have been several major attempts to solve the problem of having a computing device recognize handwritten input and today, several open source SDK's are available that can be seamlessly integrated into applications to enable accurate handwriting recognition.

We plan to utilize the powerful handwriting recognition SDK's to build an application for a phone or a tablet in which kids can draw a flowchart for given problems and the app generates a Python program for it. Also, we will have the app provide suggestions/hints for a particular problem that appears on the screen so that it keeps them engaged and not give up after a certain point of time. After the code is generated, we will have some test cases for the designed problems which would be used to provide feedback to correct their solution, if incorrect.

First of all, having this app for a tablet or phablet sized device gives the kids a bigger canvas, which is great. Further, placing objects on the canvas and writing on it using a stylus is quicker and easier rather than typing. The application can be extended such that it can generate code in other programming languages as well. This will further strengthen their knowledge of different programming languages as they can have a comparative view of the programs. Machine learning techniques can then be applied, by collecting data about common errors made by children, enabling more relevant feedbacks to their solution. We believe it would be a great educational platform for kids to start solving problems from a computational point of view and be active users of technology.

CHAPTER 3

PROJECT REQUIREMENTS

3.1 USER STORIES

As a ***Student***, I want to start learning programming using an interactive application on my tablet. I need to develop my computational thinking by drawing flowcharts for a problem and see the logical flow step by step. I also want to see how my friends are doing and compare my scores with them.

3.2 FUNCTIONAL REQUIREMENTS

Core Features

The user needs an active internet connection to use the application.

Feature	User action(s)	System requirement(s)
Facebook login	• The user must provide his Facebook login credentials.	• Authenticate and allow the user to use the application.
Tutorials	• The user must select the option to view the tutorials. • The user can choose to move to a particular example in the tutorial.	• Display a video explaining how to use the app. • Audio/Video explanation of the solution of a particular problem using the app features.
Read out questions	• Go to a particular problem.	• Read out the text displayed on the screen.
Difficulty levels	• Choose a difficulty level	• Display a list of questions corresponding the difficulty level selected.
Draw shapes	• Draw flowchart shapes on the canvas.	• Recognize the drawn shape.

Select shapes	• User can also choose to select the shapes from the pallet provided.	• Display the pallet on and enable dragging of shapes on canvas upon selection.
Write text	• Choose the text option and the shape to add the text. • The text can be in print or cursive.	• Display the text option and associate the text with the shape selected.
Delete text/shapes	• Select the text/shape to be deleted.	• Remove the shape/text from the canvas upon delete selection.
Modify connections	• The user must simply choose the arrow to connect to a different shape.	• Change connection points and the data structure mapping for the updated connection.
Clear canvas	• Tap the clear canvas button.	• Clear all contents from the canvas.
Provide hints/suggestions	• Tap the 'hints' button to see suggestions/hints while solving a particular problem.	• Display hints to help the user for solving the problem selected to be solved.
Step by step execution	• Choose to view step by step execution of flowchart. • Select next to go to next step.	• Light-up/highlight the flowchart shape being interpreted in the current step. • Display the variable values in the current execution step.

Generate code	• Tap the generate code button after finishing the flowchart.	• Generate a python code for the flowchart drawn.
Provide feedback		• Provide feedback on the solution provided by the student.
Share on Facebook	• Tap the share on Facebook button on successfully completing a question.	• Post the details on the user's Facebook wall.
Leaderboard	• User chooses to view the leaderboard to see ranking and number of points earned.	• Fetch details stored on the cloud and display the leaderboard with number of points.
Logout	• User must tap the logout button.	• End the user session and logout of the application.

Optional Features

Feature	**User requirement(s)**	**System requirement(s)**
Have machine learning to understand common errors		• Provide suggestions based on the data of common errors made on a particular problem.

3.3 NON-FUNCTIONAL REQUIREMENTS

- A highly attractive UI
- Robustness
- Security
- Ease of use
- Performance (Minimum time lag)

CHAPTER 4

DEPENDENCIES AND DELIVERABLES

4.1 DEPENDENCIES AND DELIVERABLES

We do not see a dependency now from handwriting recognition point of view. The idea of having the user draw the flowchart and write text, and have the system interpret it is possible with the help of Samsung S Pen SDK. We have already integrated the S Pen SDK with our app and have seen the shape and text recognition working seamlessly.

The following is a list of deliverables that we have planned –

1. An Android application –

   We will deliver and android application for the system described above with features like giving the user a canvas to draw flowcharts, interpreter and compiler, social media integration and collaboration.

2. Tutorial bundle –

   The application has a tutorial section which contains a videos pushed on the cloud. We will deliver the tutorial as well.

3. Set of questions –

   We plan to have a set of questions on the cloud those will pop up on the screen when the user chooses to solve a problem. We will have them delivered.

4. Publishing on PlayStore –

   We also plan to have our app published on the Play Store.

5. Release document –

   This will include the details such as privacy policy and the description of the application.

CHAPTER 5

PROJECT ARCHITECTURE

5.1 PROJECT ARCHITECTURE

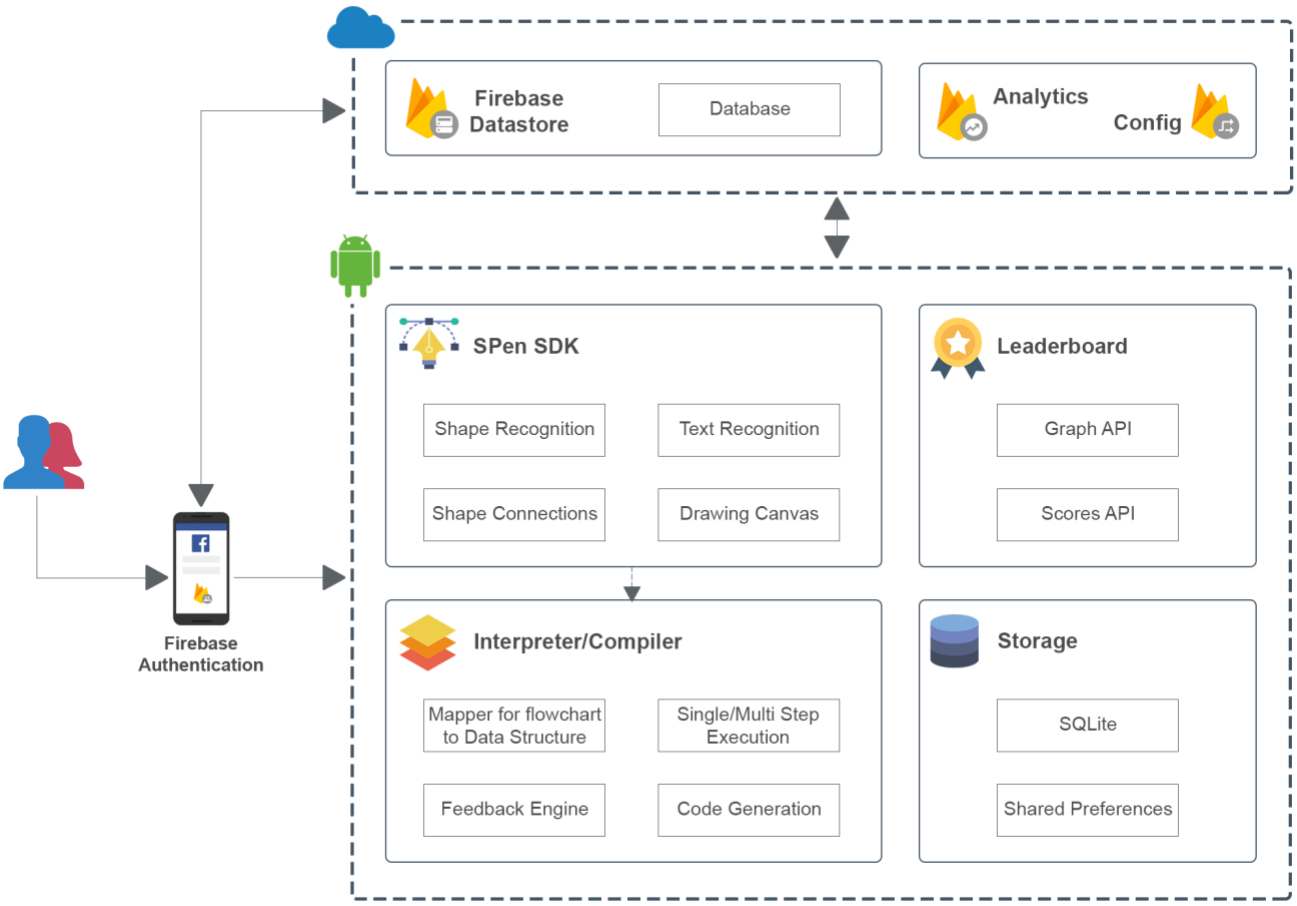


Figure 5.1.1: *High-level architecture (https://www.flaticon.com)*

The figure above shows a high-level architecture of the system where we have managed to break it down into subsystems. The architecture starts with the user having to login to the app using a social media account which happens to be Facebook here. After being authenticated, the user has access to the entire application. Now, there are several subsystems inside the application like Samsung SPen SDK, flowchart interpreter and step by step execution engine, feedback engine, code generator, the leaderboard with Graph API and a local SQLite database.

To talk about the application architecture first, we analysed MVC, MVP, and other architecture patterns. We believe that an architecture such as MVP is best for our application. The reason being, today a lot of changes made are on the UI side rather than the business logic. In an application such as ours, UI is very critical and ought to be good since the targeted users' age group is between 13 and 18. Here, since the view does not interact with the model at all, i.e. the business logic, UI updates, and fixes are easier to make. We can have a one to one mapping between the views and presenters to achieve further modularization and make unit testing

simpler. The elimination of the dependency of views on controllers and models as in MVC would be advantageous in designing complex views since classes and methods in business logic would not have to be modified when we need to add something or scale up.

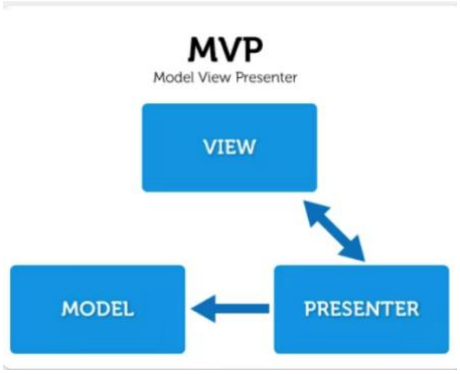


Figure 5.1.2: *MVP design (http://www.tinmegali.com/en/model-view-presenter-android-part-1/)*

A brief description of each subsystem shown in Figure 1 is below.

Authentication

We plan to use Firebase authentication for Facebook Login. The data for leaderboard that we have in the application can be stored over the cloud in the Firebase database. The details of the users' friends using the app can be pulled up using the Graph API. This will help us map data to the users' account specific information tables. A user can then view the scores of all the users of the app or only of friends on Facebook.

Samsung S Pen SDK

The Samsung S Pen SDK provides us with handwriting and shape recognition capabilities. It is the most essential subsystem given that the user will draw a flowchart on the canvas using a stylus and insert handwritten text inside the shapes. This would be our foundational support for interpreting the users' solution.

Step executor

This will again be an important subsystem in the application since the user can execute his flowchart step by step to understand the logical flow and know the values of variables at each step of execution. This will help the user improve his/her logical aptitude.

Code generator

The code generator subsystem would interpret the entire flowchart by analyzing the interconnection between all the shapes and map it to a graph data structure. In turn, this would be used to generate code in Python.

SQLite

We plan to have a local NoSQL database in the system for storing a few questions fetched from the cloud so that the experience is smooth. We can also use this to save the questions marked by the user for solving later or other purposes.

Leaderboard

The leaderboard will be a part of the application and we are treating this as a separate subsystem since it involves fetching of details from the users' Facebook account using the Graph API. It will help the users' see how their friends are doing and what is the users' ranking among them as well among all the app users'.

Feedback engine

The feedback engine will help to provide feedback to the user relevant to the particular question being attempted to solve. It will involve running some test cases and checking corner cases for the question being attempted.

The overall architecture division will help us focus on one set of features at a time and help cover dependencies for further tasks. Having a cloud support will help us scale up the system. We plan to pull out questions from the cloud which will mean efficient space utilization of the device and pushing new questions in the system without having to send out an update.

CHAPTER 6

PROJECT DESIGN

6.1 UML DIAGRAMS

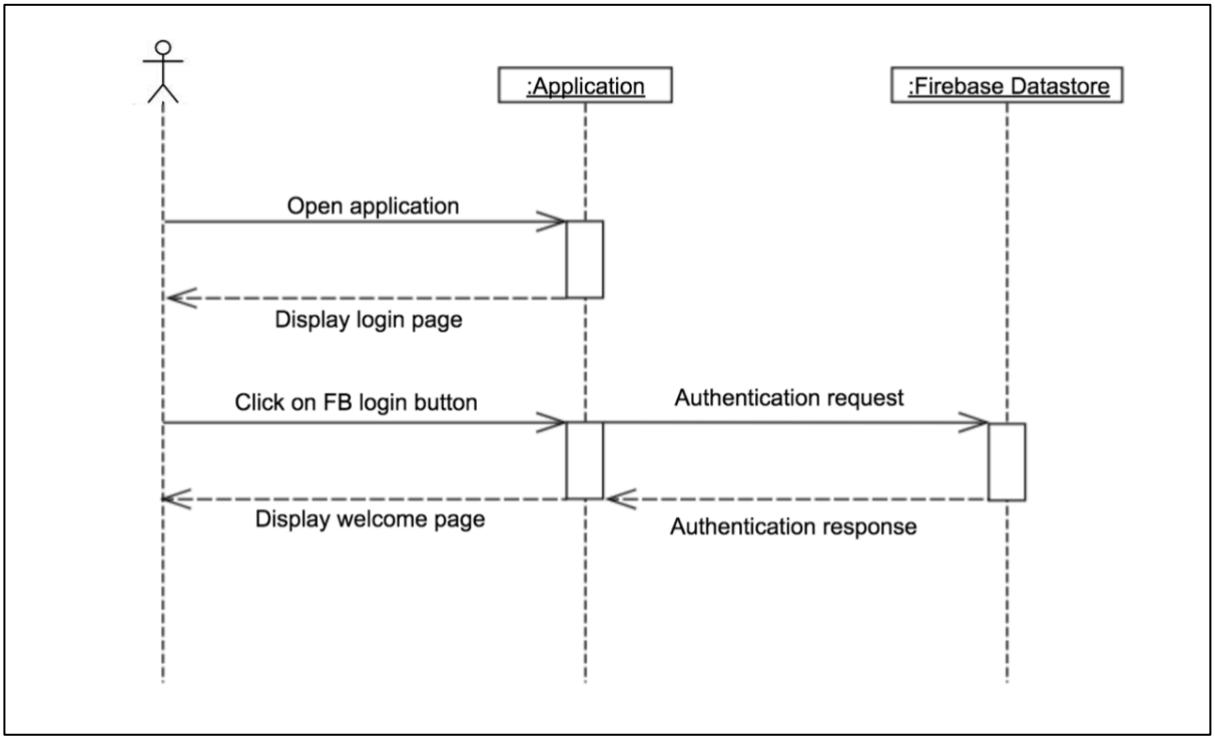


Fig. 6.1.1 – *Login sequence diagram*

The sequence diagram shown above in Fig. 6.1.1 shows the interaction between three entities, viz. user, application and firebase database. Once the user launches the application, the login page is displayed. The user then needs to login using his/her Facebook login details, for which the application sends an authentication request to the firebase database. The firebase database validates this request and sends an authentication response to the application. Upon successful authentication, the application displays the welcome page to the user.

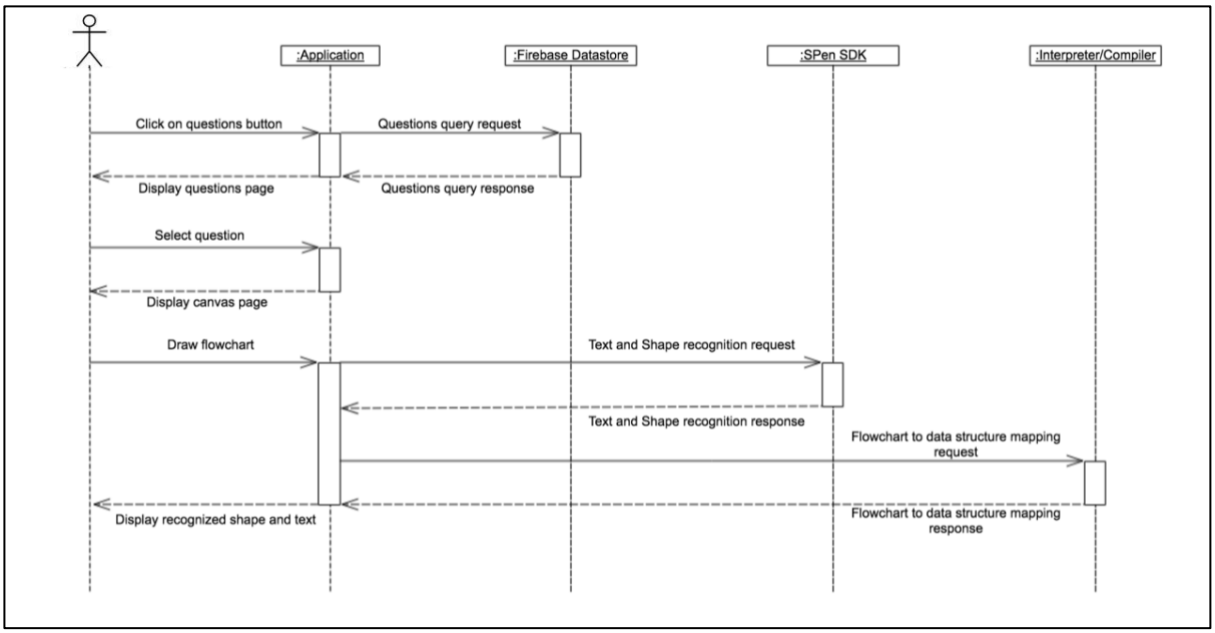


Fig. 6.1.2 – *Flowchart recognition sequence diagram*

The sequence diagram shown above in Fig. 6.1.2 shows the interaction between five entities, viz. user, application, firebase database, S Pen SDK and interpreter/compiler. Once the user click on the question button, the questions page is displayed. The user then needs to select a question to begin, and the application provides a blank canvas with a palette of options to draw or edit a flowchart. The user then using S Pen draws flowchart shapes and enters handwritten text into the shapes. The Samsung S Pen SDK converts these shapes into recognizable flowchart shapes and digital text. Once the user inputs the complete flowchart, the application sends a mapping request of flowchart to data structures to the interpreter and compiler module. This module maps the flowchart with a data structure and returns a request on success.

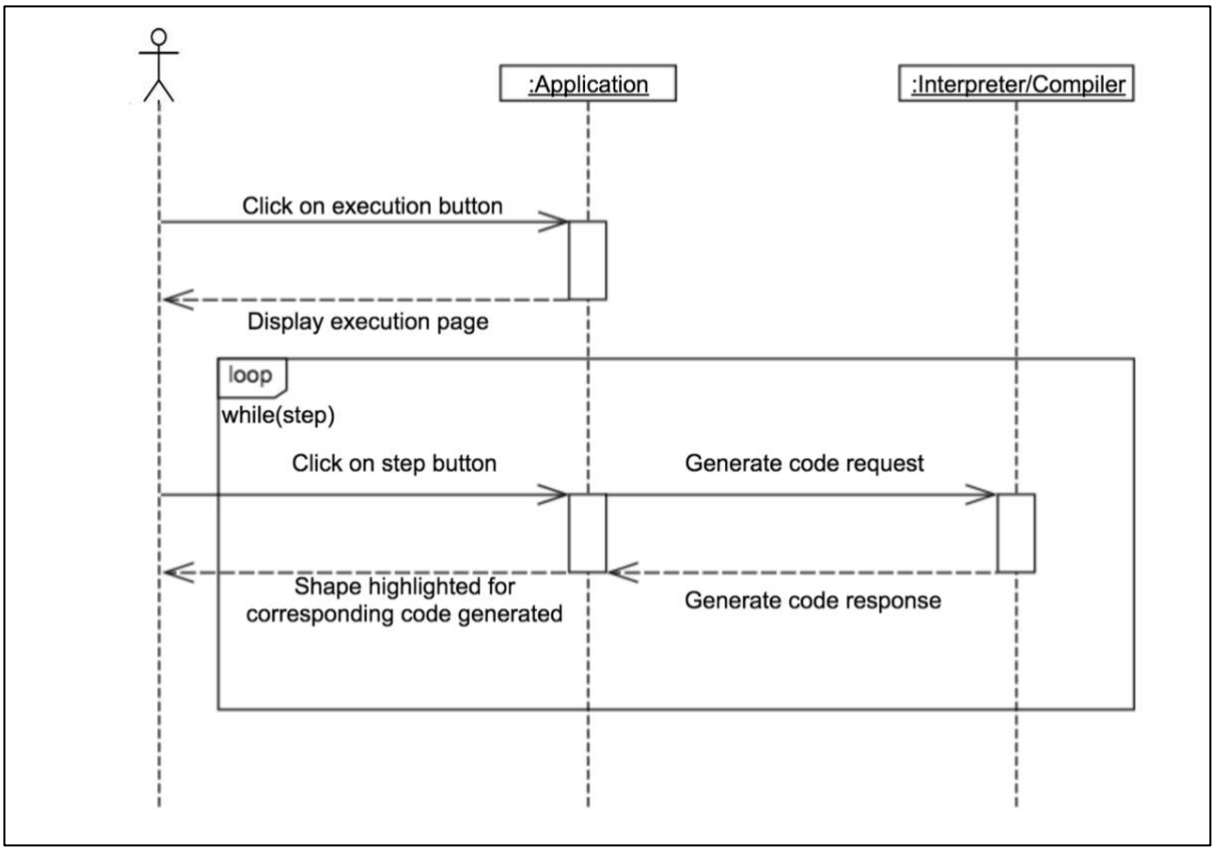


Fig. 6.1.3 *– Step execution sequence diagram*

The sequence diagram shown above in Fig. 6.1.3 shows the interaction between three entities, viz. user, application and interpreter/compiler. Once the user clicks on execution button, the execution page is displayed. Now, the user can perform one click execution or can perform step by step execution. For every click on the step button, the application sends a generate code request to the interpreter/compiler module, which returns a generate code response for the application. The application provides an animated and interactive UI by highlighting the flowcharts shapes for the current step executed.

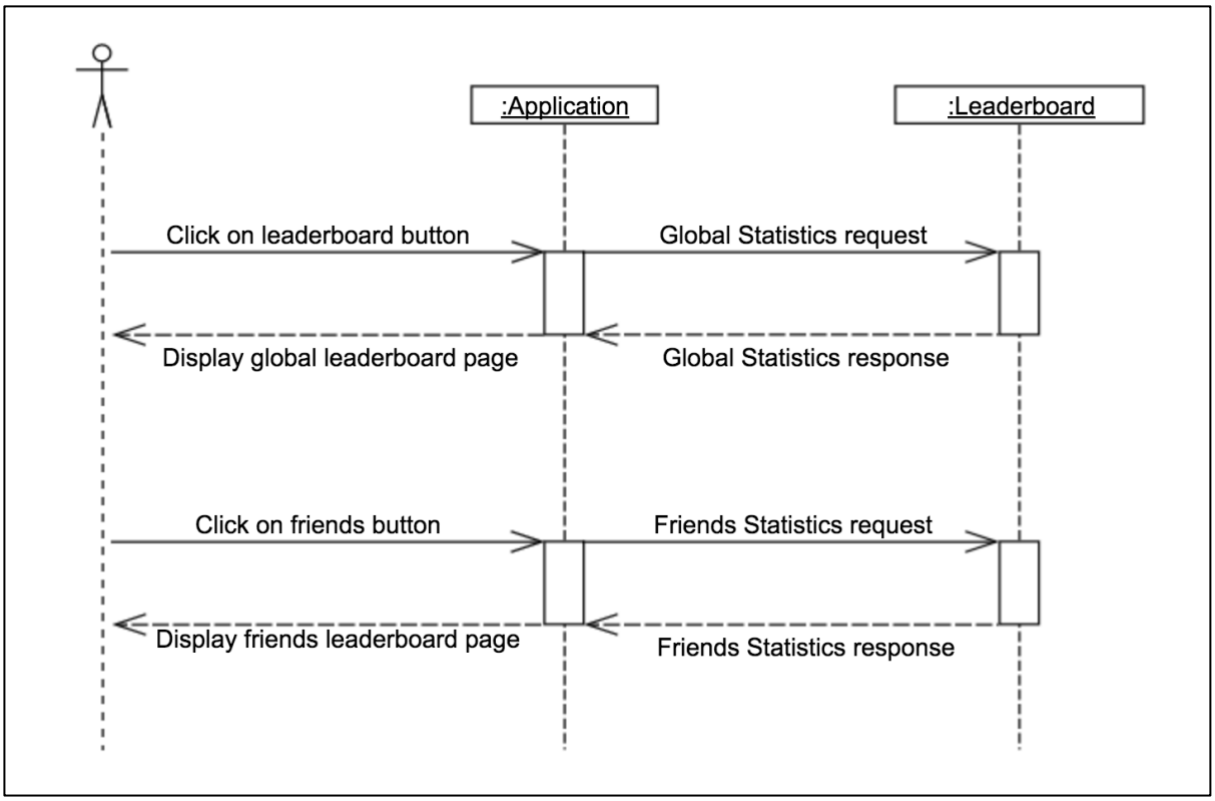
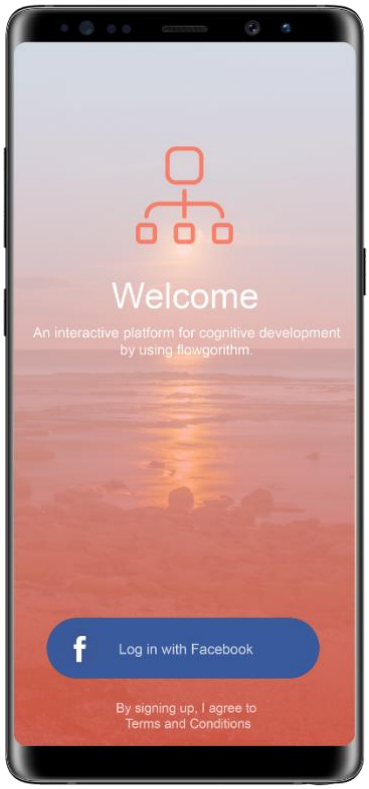


Fig. 6.1.4 – *Step execution sequence diagram*

The sequence diagram shown above in Fig. 6.1.4 shows the interaction between three entities, viz. user, application and leaderboard. Once the user clicks on the leaderboard button, the application sends a global statistics request to the leaderboard module. The leaderboard module responds with a global statistics response and the global leaderboard page is displayed. If the user now clicks on friends button on the leaderboard page, a friends statistics request is sent to the leaderboard module. The leaderboard module responds with a friends statistics response and the friends leaderboard page is displayed.

6.2 UI MOCKUPS



The login screen is shown in the figure. Login will be using Facebook credentials through Firebase authentication. The user will be granting the access to fetch user's public profile and friends.

Figure 6.2.1: *Login screen (https://www.uplabs.com)*

The drawing canvas is one of the main screens in the application. It leverages the SPen SDK and enables the user to draw shapes, connections, and text on the canvas that will be recognized to obtain a well-structured and designed flowchart. The shapes and text can be deleted or modified. Also, the connections between the shapes can be modified.

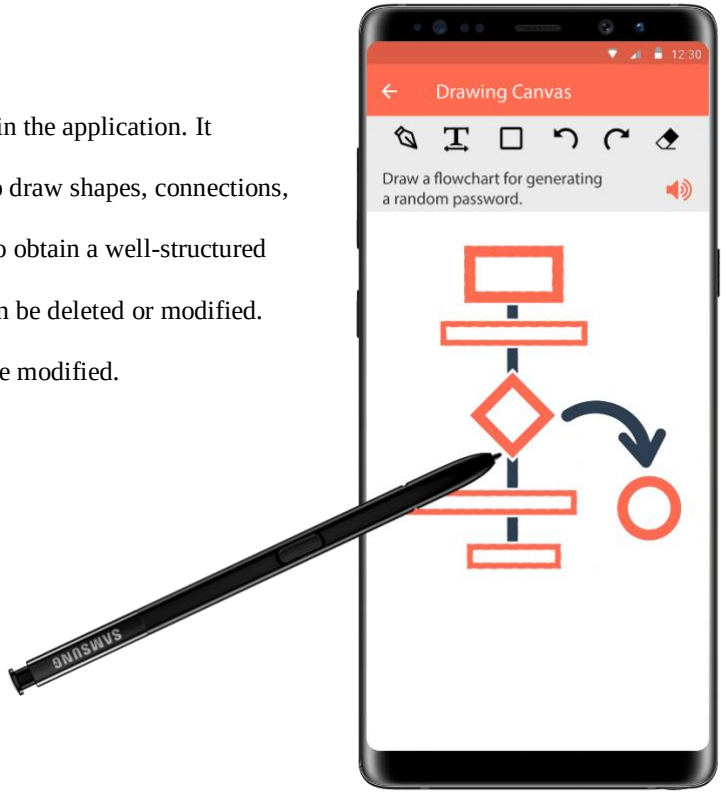


Figure 6.2.2: *Drawing Canvas (https://www.uplabs.com)*

The flowchart will be interpreted within the application itself and then a python script will be generating based on the flowchart which could be compiled and executed inside the application as shown in the figure.

Figure 6.2.3: *Program code screen (https://www.uplabs.com)*

The leaderboard screen will be showing the Global list as well as friends list of the highest scorers for the application. This module will be using the Facebook Graph API for fetching the friend's information and then will form the leaderboard list.

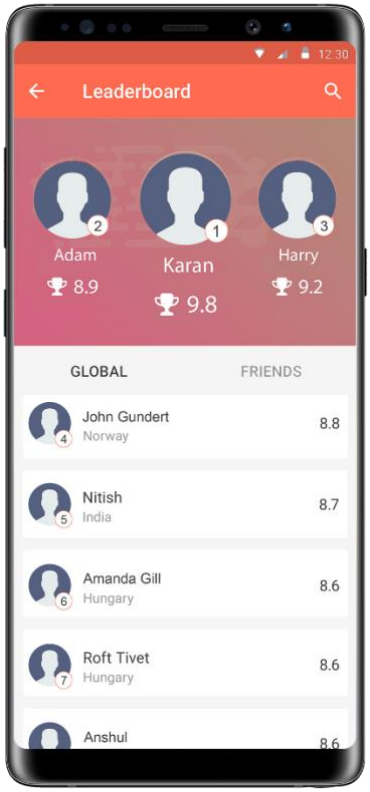


Figure 6.2.4: *Leader-board Screen (https://www.uplabs.com)*

6.3 DATA DICTIONARY

Login Document

Field Name	Data Type	Field Length	Constraints	Description
ID	INT	11	Auto Increment	Auto Generated sequence number
EMAIL	VARCHAR	522	Primary Key	User's unique email address
LAST LOGIN	DATETIME		Not Null	Last login date and time

Tutorials Document

Field Name	Data Type	Field Length	Constraints	Description
ID	INT	11	Auto Increment	Auto Generated sequence number
VIDEO	TEXT		Not Null	Video file location on Firebase storage
VIDEO NAME	VARCHAR	522	Not Null	Video file name

Problems Document

Field Name	Data Type	Field Length	Constraints	Description
ID	INT	11	Auto Increment	Auto Generated sequence number
PROBLEM	VARCHAR	522	Primary Key	Problem statement
PROBLEM NAME	VARCHAR	522	Not Null	Problem Name

PROBLEM CATEGORY	VARCHAR	522	Not Null	Problem's category
SOLUTION	TEXT		Not Null	Solution image location on Firebase storage
SOLUTION AUDIO	TEXT		Not Null	Solution audio location on Firebase storage

Progress Document

Field Name	Data Type	Field Length	Constraints	Description
ID	INT	11	Auto Increment	Auto Generated sequence number
EMAIL	VARCHAR	522	Foreign Key	User's unique email address
SCORE	VARCHAR	522	Not Null	User's overall score
LEVEL	VARCHAR	522	Not Null	Level of difficulty that the user is currently solving

CHAPTER 7

QA, PERFORMANCE, DEPLOYMENT PLAN

7.1 INITIAL RELEASE TEST PLANS

For the initial release, the following changes/implementations are planned to each sub-system or component.

Table – Initial Release Software Changes/Implementations

Subsystem	Change/Implementation Detail	Level of Change *	Regression Test Plan
Firebase Authentication	Firebase Setup Authentication Settings	High (Initial Release)	Full
SPen SDK Integration	Drawing Canvas Shape Recognition Text Recognition Shape Connections	High (Initial Release)	Full
Interpreter/ Compiler	Machine learning model for recognition and prediction	High (Initial Release)	Full
Cloud Integration	Firebase Setup Database Schema Design Remote Functions	High (Initial Release)	Full
Leaderboard	Facebook Graph API Leader selection	High (Initial Release)	Full
Local Storage	Database Schema Design Utility Functions	High (Initial Release)	Full

*where high is > 50% code changed, medium is 10%-50% code changed, and small is < 10% code change

All changing components for the initial release will run unit tests as part of the build process; failing unit tests will cause the build to fail and not be released for testing. Some components may also leverage white box functional testing as part of the build process. Again, functional test failures will cause the build to fail and not be released for testing.

Component Level Test Plan

Firebase Authentication Test Plan

Verification of the authentication will be done by checking if the user is authenticated correctly and the information fetched from the Facebook API is valid. The correctness of authentication will be checked by comparing the obtained parameters with the expected output during Verification & Validation (V&V).

SPen SDK Integration Test Plan

SPen SDK integration will be verified by performing a subset of tests (manual & automated) for regression purposes. The tests will have pre-defined output data stored in external object repository which will be used to verify the generated output from the integration process.

Interpreter/Compiler Test Plan

Performance testing will be done on the interpreter/compile module using different datasets. 10% data from the initial set will be taken out as the testing set for V&V. Error analysis for the interpreter/compiler will be done using the different data set.

Cloud Integration Test Plan

Cloud Integration will be tested by verifying the data transfer process between the application and cloud storage. Because all the Cloud Integration test cases are automated, all will be run for the V&V effort.

Leaderboard Test Plan

The leaderboard will be verified by sorting the user scores list and then comparing it to the actual leaderboard list in the application. The score data will be fetched using the Facebook Score API or the Achievements API.

Local Storage Test Plan

Local storage will be tested by checking the working of utility functions (setters and getters) that are used to communicate with the local SQLite database. All the unit tests for the local storage will be run during the development phase.

Integrated System Testing

The component testing section described the test plan for each of the changed components of the release. Releases must be verified for compatibility with the current infrastructure before a release is delivered to production. In order to verify this seamless integration of all components, changed and unchanged, integrated system testing will also be performed during V&V. This will primarily include running of designated workflow system tests that are meant to exercise anticipated case workflows throughout the system.

The changed components for the initial release as identified in Table above will undergo full risk related regression testing; all test cases that are linked to requirements and defects which are linked to risk items will be run for V&V.

CHAPTER 8

IMPLEMENTATION PLAN & PROGRESS

The implementation plan is divided into the following phases:

1) Initiation phase **Completed**

   This phase includes the project discussion, abstract proposal, and literature search for handwriting recognition. The proposal was prepared with a thorough research of the topic which was then backed by substantial literature search from various research papers and publications. This phase is 100% completed.

2) Definition Phase **Completed**

   The second phase mainly included creating user stories, standard compliance project requirements, functional requirements and non-functional requirements of the project. This phase is 100% completed.

3) Design phase **Completed**

   The third phase has high-level architecture which involves the design of the android application and its interface with other systems such as interpreter and compiler module, database module, cloud module and firebase module. This is followed by detailed architecture, UI mockups, UML diagram preparation. This phase is 100% completed.

4) Development phase **In Progress**

   The development phase will include the environment setup for implementing the project and integrating the all the modules. The development phase also includes the development of application code to identify a structured flowchart and text associated with its blocks and mapping it with data structure, build a code interpreter and code generator and, the integration of SQLite and Samsung S Pen SDK in the android application. This phase is initiated and will be completed as per the project timeline.

5) Implementation phase **Not Started**

   The implementation phase involves implementation of developed android application, code interpreter, code generator, cloud and database interface in the system. This phase is not started will be performed in the next semester where implementation will be majorly considered.

6) Follow up Phase **Not Started**

   The last phase includes the verification, validation and testing. The verification, validation and testing will be done for the functional features of the mobile application, local and cloud database interface, code interpreter, code generator. This will provide the efficiency and accuracy of the system.

CHAPTER 9

PROJECT SCHEDULE

9.1 GANTT CHARTS

Figure 9.1.1: *Gantt Chart*

9.2 PROJECT SCHEDULE WITH DEPENDENCIES AND TASK ASSIGNMENT

Task Name	Start	Finish	Assigned To	Dependencies	Comments
Initian Phase	**08/25/17**	**09/20/17**			**Completed**
Project Discussion	08/25/17	08/29/17	Karan, Nitish, Nrupesh, Anshul		Completed
Abstract Proposal	08/30/17	09/07/17	Karan, Nitish, Nrupesh, Anshul	2	Completed
Literature Search	09/08/17	09/20/17	Karan, Nitish, Nrupesh, Anshul	2, 3	Completed
Definition Phase	**09/08/17**	**09/26/17**			**Completed**
User Stories	09/08/17	09/11/17	Nitish, Nrupesh	3	Completed
Project Requirements	09/21/17	09/26/17	Karan, Nitish, Nrupesh, Anshul	3, 4, 6	Completed
Functional Requirements	09/21/17	09/26/17	Karan, Anshul	3, 4, 6	Completed
Non-functional Requirements	09/21/17	09/25/17	Nitish, Nrupesh	3, 4, 6	Completed
Design Phase	**09/27/17**	**10/24/17**			**Completed**
High-level architecture	09/27/17	09/29/17	Karan, Nrupesh	3, 6, 7	Completed
Detailed architecture	10/02/17	10/16/17	Nitish, Anshul	7, 11	Completed
UI Mockups	10/17/17	10/19/17	Karan, Nrupesh	11, 12	Completed
UML diagrams	10/17/17	10/24/17	Karan, Nitish	11, 12	Completed
Other sketches	10/17/17	10/19/17	Nitish, Anshul	11, 12	Completed
Development Phase	**10/17/17**	**11/17/17**			**Completed**
Environment setup	10/17/17	10/20/17	Karan, Nitish, Nrupesh, Anshul	11, 12	Completed
Configuration of local and cloud database	10/23/17	10/25/17	Nitish, Nrupesh	12, 17	Completed
Integration of Samsung SPen SDK in our application	10/23/17	10/30/17	Karan, Nrupesh	17	Completed
Harnessing the power of handwriting recognition of SPen SDK	10/31/17	11/03/17	Nitish, Anshul	19	Completed
Identifcation of shapes of flowchart	11/06/17	11/09/17	Karan, Nitish	19, 20	Completed
Association of text and arrows with flowchart blocks	11/10/17	11/17/17	Karan, Nrupesh	20, 21	Completed
Implementation Phase	**11/20/17**	**03/07/18**			**In Progress**
Development of android application to identify structured flowcharts	11/20/17	12/04/17	Nrupesh, Karan	19, 20, 21, 22	In Progress
Mapping flowchart to data structures	01/15/18	01/23/18	Nitish, Anshul		Not Started
Implement step executor	01/24/18	02/12/18	Nitish, Karan	24, 25	Not Started
Implement code generator	02/13/18	02/20/18	Nrupesh, Anshul	24, 25, 26	Not Started
Implement interactive feedback engine	02/21/18	02/28/18	Karan, Nitish	24, 25, 26, 27	Not Started
Implement leaderboard	03/01/18	03/07/18	Anshul, Nrupesh	24, 25, 26, 27, 28	Not Started
Follow up Phase	**03/08/18**	**05/08/18**			**Not Started**
Verification, Validation and Testing	03/08/18	04/30/18	Karan, Nitish, Nrupesh, Anshul		Not Started
VVT for android application features	03/08/18	03/20/18	Nitish, Anshul	29	Not Started
VVT for flowchart to data structures mapping	03/21/18	03/28/18	Karan, Nrupesh	32	Not Started
VVT for step executor	03/29/18	04/05/18	Nrupesh, Anshul	32, 33	Not Started
VVT for code generator	04/06/18	04/13/18	Nitish, Karan	32, 33, 34	Not Started
VVT for interactive feedback engine	04/16/18	04/24/18	Nrupesh, Anshul	32, 33, 34, 35	Not Started
VVT for leaderboard	04/25/18	04/30/18	Karan, Nitish	32, 33, 34, 35, 36	Not Started
Project report and demo	04/15/18	05/08/18	Karan, Nitish, Nrupesh, Anshul		Not Started

Figure 9.2.1: *Project Schedule with dependencies and Task Assignment*

CodeWizard – An Interactive Platform to Learn Computer Programming

Project Name

Cloud Integration module

Functional Specification

Version 1.0

Nitish Potdar
(nitish.potdar@sjsu.edu)

Table of Contents

1. Version History

Version	Changes
1.0	Initial document

2. Introduction

We aim to develop an interactive platform for naive programmers to learn programming algorithms by drawing flowcharts. The flowchart will be converted into executable code and an animated UI will help programmers to understand the flow of program logic.

The high-level architecture is as shown below in Fig. 2.1.

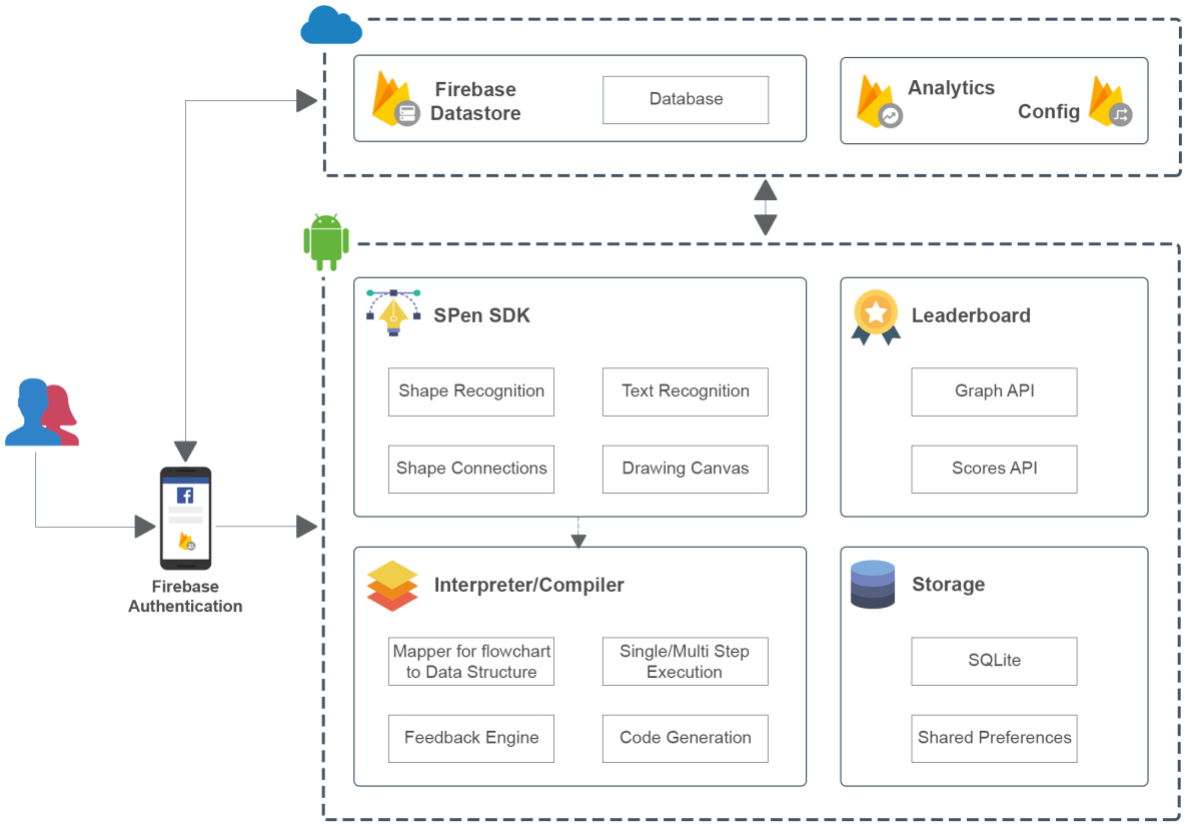


Fig. 2.1 – High level architecture

The architecture consists of various modules such as:

- **S Pen SDK module:** This module provides the user with a drawing canvas and question sets. It converts the handwritten shapes into recognizable flowchart blocks. It will also convert the handwritten text into digital format and associate the corresponding text with the flowchart blocks.

- **Interpreter/Compiler module:** This module is responsible for the code generation and step by step execution in the system. Once the user inputs the flowchart on the canvas, it will be mapped to a data structure by the mapper, then code generator will generate

the code. Single/multiple step execution unit will be responsible for the execution of the code in steps or all the steps in a single click. This module also consists of a feedback engine, which will interact with the S Pen SDK module to provide an interactive feedback to the user by highlighting the blocks during step by step execution.

- **Cloud integration module:** This module is the Firebase cloud implementation. It will be responsible for features such as authentication, database storage, analysis of user skillset and preferences.

- **Leaderboard module:** This module will use the graph API and scores API to acquire the user data, create and compare rankings, and display them on the leaderboard.

- **Local Storage module:** This module stores some questions in a local database, so that they are available without internet connection. Once the user marks questions for offline access, they will be downloaded from the Firebase database and stored locally in the SQLite database.

The advancement in cloud technology provides us with a capability to harness this domain. By integrating cloud in our application, we can drastically decrease the application size. To teach the students with all the programming algorithms, there is a need to have adequate question and solution set. We can achieve this by implementing cloud infrastructure in our application. Cloud analytics will help our application to perform user preferences analysis using cloud computing. This will help us to provide the user with appropriate level of questions set. The cloud configuration will help us to update the question set offline without the need to download an application update. Thus, this will enrich the user experience and motivate him/her to focus on learning the algorithms as per his/her necessities. We will be utilizing Firebase services to implement the storage, authentication, database, notifications and remote config features in the application.

3. References

Workbook 1 submitted on Canvas contains the architectural description of the project

Google Developers (N.D.) *Firebase*. Retrieved from https://firebase.google.com

Osman A., El-Refaey M. & Elnaggar A., "Towards Real-Time Analytics in the Cloud," *2013 IEEE Ninth World Congress on Services*, Santa Clara, CA, 2013, pp. 428-435. doi: 10.1109/SERVICES.2013.36

4. Requirements

As stated above, we will be using Firebase cloud services to implement cloud interface into our application. The functional requirements of the cloud integration module are-

1. Firebase analytics service can perform user preferences analysis and generate appropriate set of questions for each individual user
2. Firebase remote configuration service can update the application UI remotely
3. Firebase database service can be updated as per the need and is integrated with the application to reflect the changes without need to download an application update
4. Firebase notification service can send notifications to targeted users for improved and interactive experience
5. Firebase authentication service supports the application with user account validations and helps track the application usage for users

The non-functional requirements of the cloud integration module are-

1. Fast user authentication
2. Less response time for requests sent by/to the application
3. Database updation is simple and easy
4. Availability of Firebase services without any intervene
5. Database schemas match with the local database storage

5. Functional Overview

The cloud interface module provides the flexibility to our application. It will also be easy to scale as per demand and update the coursework on the go without the need of downloading an update of the application by the user. This will help the teachers in the coursework as they can update the questions in the Firebase Console and the application will automatically receive an updated set of question from the database. The cloud interface module will use the facilities of cloud computing keeping the RAM usage low on the device.

Fig. 5.1 – Cloud Integration module (Firebase module)

The Firebase module shown above in Fig. 5.1 depict the implementation of Firebase database, Firebase analytics and Firebase remote config services in the system.

5.1 Configuration/ External Interfaces

- External Interfaces
    - SQLite
    - Leaderboard module
    - Interpreter/Compiler module
    - S Pen SDK module

- Dependencies
    - User account is properly setup
    - Good internet connection to provide reliable and fast responses
    - Database schemas match the application setters and getter messages

6. Implementation

To implement cloud interface module, we first need to import the Firebase package in the application code. So, whenever the application is started, an instance of the cloud interface is created over which the application can perform the activities related to cloud analytics and cloud configurations. All the services can be monitored by using the Firebase console.

We tend to design and implement a database which can store the questions and solutions categorized into easy, medium and hard levels. A leaderboard module will also be implemented to track the scores and maintain global and user statistics. The data needed for this module will be stored in the Firebase database in JSON format. The Firebase console helps to change the behaviour and appearance of the application using the Firebase remote config service. The in-app default values are replaced by the values from the Firebase console. To implement the Firebase analytics service, we need to integrate Firebase analytics SDK in our application and the service can be handled using the Firebase console. And to implement the Firebase authentication service, we need to integrate Firebase authentication SDK in our application.

Thus, for every service that needs to be implemented, we need to integrate the Firebase service SDK in the application and the service can be controlled from the Firebase console. The Firebase database stores the questions and their solutions in text and audio format. It will also store the application tutorial in the Firebase database in video format.

The database schema for storing the question set is-

Field Name	Data Type	Field Length	Constraints	Description
ID	INT	11	Auto Increment	Auto generated Sequence Number
PROBLEM	VARCHAR	522	Primary Key	Problem Statement
PROBLEM NAME	VARCHAR	522	Not Null	Problem Name
PROBLEM CATEGORY	VARCHAR	522	Not Null	Problem Category
SOLUTION	TEXT		Not Null	Solution image location on Firebase storage
SOLUTION AUDIO	TEXT		Not Null	Solution image location on Firebase storage

Table 6.1 – Problem database schema

The database schema for storing tutorial is-

Field Name	Data Type	Field Length	Constraints	Description
ID	INT	11	Auto Increment	Auto generated Sequence Number
VIDEO	TEXT		Not Null	Video file location on Firebase storage
VIDEO NAME	VARCHAR	522	Not Null	Video file name

Table 6.2 – Tutorial database schema

The database schema for login authentication is-

Field Name	Data Type	Field Length	Constraints	Description
ID	INT	11	Auto Increment	Auto generated Sequence Number
EMAIL	TEXT	522	Primary Key	User's unique email address
LAST LOGIN	DATETIME		Not Null	Last login date and time

Table 6.3 – Login database schema

7. Testings

7.1 General Approach

To test the working of this module, we need to first integrate the Firebase SDKs in our application and test the services using the Firebase console. Firebase console acts as the services monitoring and controlling unit. Thus, as a general-purpose test, we can test the working of the services by invoking them from Firebase console and monitoring them in the application.

The data availability from cloud storage to all the module need to be tested by interfacing each module with the Firebase database independently. Once all the data connectivity with all the modules is tested, we can test the data flow in the system is working seamlessly. To provide accurate and appropriate results, we need to test the individual units in the Firebase cloud module. These tests have been documented in the next section.

Testing of the cloud integration module can be done by using the Selenium framework. The automation scripts will generate reports which can be evaluated to check for any discrepancies in the data flow between the cloud module and the other modules in the system.

7.2 Unit Tests

Below is the list of unit tests that can be performed on the sub-tasks or sub-units in the module by giving them input separately-

- Firebase Authentication Service
    - Check that the integration of Facebook API authentication using the Firebase authentication service is working
    - Check the service by providing valid and invalid combinations of user ID/password

- Firebase Database Service
    - Validate that the application can access the questions stored in the database
    - Validate that the application can access the updated question in the database without the need of downloading an update of the application
    - Validate the database stores the global leaderboard and user leaderboard details according to the database schemas

- Firebase Analytics Service
    - Validate that the user preferences are stored in the database using the analytics service
    - Validate that the user experience is interactive based on the results of analytics service

- Firebase Notification Service
    - Notifications are popped-up in the application by the notification service invoked from the Firebase console

- Firebase Remote Config Service
    - Validate that the changes made in Firebase console change the behavior and appearance of the application without the need of downloading an update of the application

CodeWizard – An Interactive Platform to Learn Computer Programming

Functional Specification

Version 1.0

Nrupesh Patel
(nrupesh.patel@sjsu.edu)

Table of Contents

Version History

Version	Changes

1. Purpose

This document describes the functional requirements, implementation overview, dependencies, and testing approach for SPen SDK integration and shape/text recognition functionality of the application.

2. Introduction

The SPen SDK is the main component that will be enabling the application to allow use of stylus or fingers for drawing the flowcharts and writing text in the respective shapes. Mobile application would be having the capability of sketching down shapes and text on the provided canvas, recognize the shapes/text with the help of SPen SDK and then connect the shapes for further interpretation. SPen SDK module will be doing various operations like shape recognition, text recognition, shape-text association, and shape connections. The input to the SPen SDK module will be input strokes using SPen or fingers and the output will be a well-structured flowchart. The main objective of SPen SDK integration is to enable shape and text recognition so that the user can draw flowcharts by having an intuitive feel of writing with hand and not typing or drag and drop.

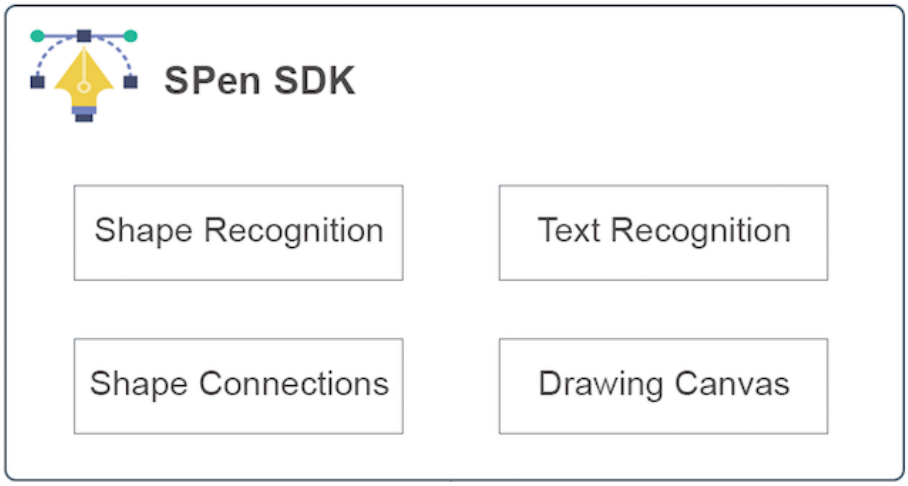


Figure 1: *SPen SDK*

3. References

Samsung Developers. (2017). *Pen SDK Programming Guide – Full Version*. Retrieved from http://developer.samsung.com/galaxy/pen/guide

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, *24*(4), 509-522.

Artieres, T., Marukatat, S., & Gallinari, P. (2007). Online handwritten shape recognition using segmental hidden markov models. *IEEE transactions on pattern analysis and machine intelligence*, *29*(2).

4. Requirements

4.1 Functional Requirements

4.1.1 Core Features

Feature	**User requirement(s)**	**System requirement(s)**
Draw shapes	• Draw flowchart shapes on the canvas.	• Recognize the drawn shape.
Select shapes	• User can also choose to select the shapes from the pallet provided.	• Display the pallet on and enable dragging of shapes on canvas upon selection.
Write text	• Choose the text option and the shape to add the text.	• Display the text option and associate the text with the shape selected.
Delete text/shapes	• Select the text/shape to be deleted.	• Remove the shape/text from the canvas upon delete selection.
Modify connections	• The user must simply choose the arrow to connect to a different shape.	• Change connection points and the data structure mapping for the updated connection.

4.2 Non-Functional Requirements

- Intuitive UI
- Ease of use
- Minimum time lag
- Efficient Recognition

5. Functional Overview

5.1 Input

The input to SPen SDK module will be strokes from either stylus or fingers. User will be interacting with the canvas provided in the application. After the user inputs any stroke, application will recognize the shape/text with the help of SPen SDK.

5.2 Processing

The SPen SDK integration will enable shape and text recognition on the canvas. The shapes will be connected using the connection lines through magnetic points on the shapes. There will be two types of recognition:

1. ***Shape Recognition***
2. ***Text Recognition***

5.3 Output

The output from the SPen SDK integration will be a well-structured and designed flowchart with corresponding text. The flowchart will be then provided to the interpreter/compiler unit.

5.4 External Interfaces

5.4.1 User Interfaces

The main user interface relating to SPen SDK is the drawing canvas. The user will be giving input strokes on the canvas which then will be translated to shapes and text based on the recognition capabilities. Those shapes would then be connected through the magnetic connection points. The problem statement will be displayed on the same screen as the canvas and will have the capability to play the audio relating to the problem. This screen will have various options to select shapes from pallet, undo and redo the input strokes, delete the shapes, text and connection and also clear the entire canvas.

5.4.2 Hardware Interfaces

Samsung Galaxy Note series is required to use SPen with the mobile application. Other devices will have to be operated with fingers.

5.4.3 Communication Interfaces

The application would be communicating with cloud services to get the user data from the external database. It will also have internet communications with the API that interacts with the cloud service to fetch the problem statements.

5.4.4 Design Constraints

The Pen package can be used on any Android device, but the Galaxy Note series devices offer more functions when we use the S Pen stylus. We can use the Pen package on devices with Android 4.0 Ice Cream Sandwich or higher. Non-Samsung devices will not be supporting the use of SPen.

5.5 Debug

Logging is implemented in various sections of the application, so that the logged information can be used when the app crashes or experiences any issue. Analytical events are also logged on various actions in

the application for further analysis and enhancements. The main events that are logged in the SPen SDK module are input stroke, shape recognized, text recognized and shape connected.

6. Implementation

The SPen SDK will be integrated first with the mobile application. After the integration with the application is completed, the recognition will be applied to user input strokes. Recognition will be having two types as shown in the chart below. After both shapes and text have been recognized, the association between them will be logged. Finally, the shapes will be connected with each other using magnetic connection points. The application implements the following feature (Samsung Developers, 2017):

- Button for inserting shape object and lines.
- When this button is clicked, a popup view allows users to select the shape/line type to insert.
- Custom mode to add shape object using input strokes.
- Touch listener for various input events.
- Shape insertion and SpenPageDoc and viewport update.
- Add SpenContextMenu to set properties of shape/line object
- When properties context menu clicked, the properties dialog will be show allows user change some properties of shape/line object.

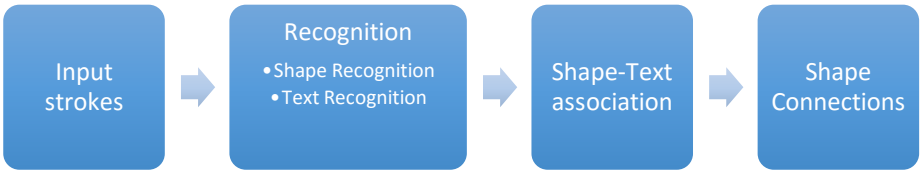


Figure 2: *Implementation Flow*

7. Testing

7.1 General Approach

The tools and frameworks that would be required to test the mobile application are:

- Appium
- JUnit
- Jenkins
- JIRA
- Samsung Galaxy Note series emulator or real device

Automation script will be executed using the continuous integration system Jenkins on timely intervals.

That script will use Appium and JUnit framework to test the mobile application and log the results into

JIRA an issue tracker tool.

7.2 Unit Tests

Following is the list of unit tests performed:

TC_001	
Function name	isSPenSupported( )
Test function name	testIsSPenSupported( )
Description	Tests if the SPen support check function shows proper responses for valid and invalid cases.

TC_002	
Function name	createSurfaceInline( )
Test function name	testCreateSurfaceInline( )
Description	Tests if the SPen surface is created properly.

TC_003	
Function name	captureSurface( )
Test function name	testCaptureSurface( )
Description	Tests if the SPen surface is captured properly.

TC_004	
Function name	recognizeShape( )
Test function name	testRecognizeShape( )
Description	Tests if the shape is recognized from the input strokes.

TC_005	
Function name	recognizeText( )
Test function name	testRecognizeText( )
Description	Tests if the text is recognized from the input strokes.

TC_006	
Function name	associateShapeText( )
Test function name	testAssociateShapeText( )
Description	Tests if the shape is associated with corresponding text.

TC_007	
Function name	connectShapes( )
Test function name	testConnectShapes( )
Description	Tests if the shapes are getting connected properly on the magnetic connection points.