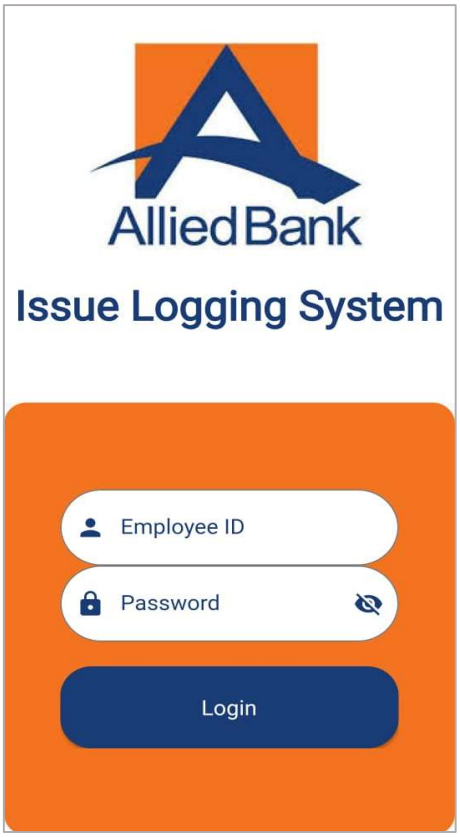
Development Details Document Issue Logging System – Allied Bank Limited

Group 24

Aadam Nadeem, Maleeha Masood, Malik Ali Hussain, Muhammad Raheem Zafar and Shahrukh Kemall



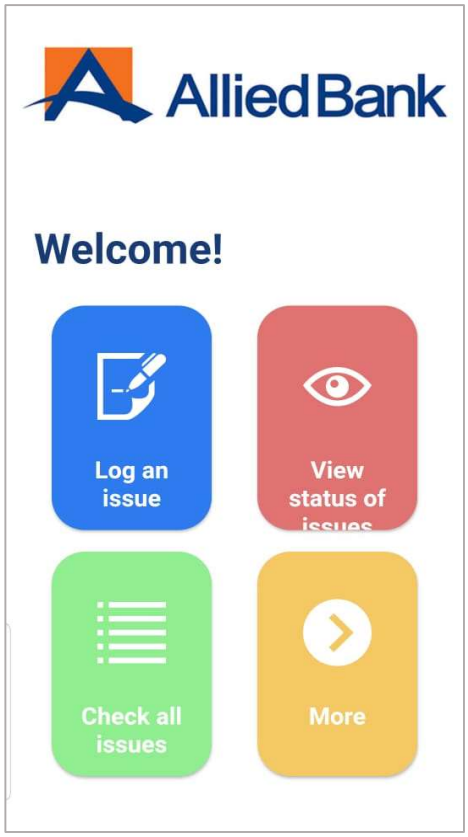


TABLE OF CONTENTS

Section 1 Overview

1.1 Document Purpose

The purpose of this document is to enlighten the user about the functioning of our developed mobile application- Issue Logging System. It maps the Software Requirements Specification Document onto our end product and allows the reader to gain insight of the work that the team has done and all the accomplishments that they have achieved.

1.2 Product Summary

As a quick refresher, the product in discussion is a mobile application prototype, initially intended to be a cross mobile platforms application through which employees at Allied Bank Limited can find ease in getting technical issues from a very broad category, ranging from simple software issues to hardware replacements, resolved. This can be achieved by logging the issue through the application after which certain already- established support employees can choose to resolve these issues.

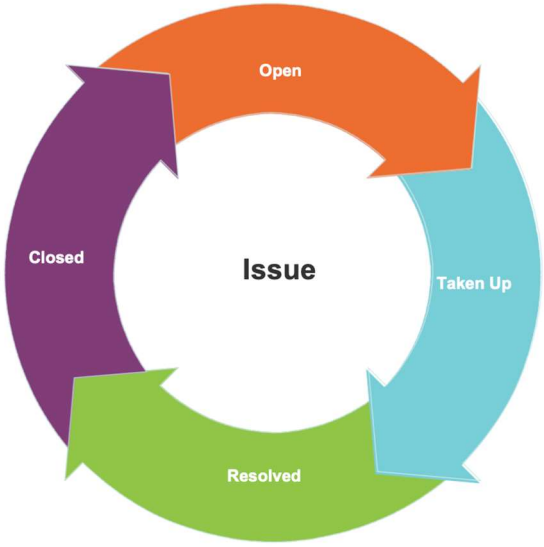
For complete understanding of the functionality of the application, it is crucial to understand a) the key roles involved and b) the growth of an issue.

a) There are five roles into which the employees at ABL have been divided: Initiator, Support Team Member, Manager, Manager and Manager.

Initiator	Any employee that can log an issue.
Support Team Member	Specialized employee that can resolve a specific category of issues.
Manager	The manager of a support team that is a set of support team members.
Monitor	Uses the app for monitoring purposes.
Admin	Has the administrator role in the application.

Note that any role can take the role of an initiator as all of them can log issues. Kindly refer to the Software Requirements Specification Document for more information on the roles.

b) An issue lives till the initiator of the problem is completely satisfied with its resolution. The lifecycle of an issue is as follows:



Note that a satisfied issue can break the cycle at closed and does not necessarily have to be in the open stage again.

A logged issue that has not been chosen by a support team member is open. An issue chosen by or assigned to a support member is taken up. Once the issue is completely solved by the support member, it is marked as resolved by them. Finally, if the initiator is satisfied, they can close the issue. Otherwise they can open it again.

Both the initiator and support team member can mark an issue as open. Only an initiator can mark an issue as closed. Only a support team member (or similar roles like manager of the support team) can take up an issue.
Also note that only a support team member or a manager can choose to resolve an issue.

1.3 Intended Audience and Reading Guidelines

This document is intended for the LUMS CS360 course staff and our client- ABL. However, any reader interested in following the development of our project can read through the document, but it is recommended to go through our previous

documentations first. Readers interested in using the app may also go through this document.

The course staff should read the complete document.
The client and readers following the progress of our project should avoid reading sections 5 and 6.
Readers interested in using the application should read the last section only after perhaps reading section 1.4.

1.4 Terminology

- ABL: Allied Bank Limited.
- Admin: Designated employees that have the administration role in the ILS system.
- API: Application Program Interface
- apk: Android Package (apk) is the package file format used by the Android operating system for distribution and installation of mobile apps, mobile games and middleware.
- Area: The broader category of a problem. For example: IT is the area of the Hardware category.
- Category: The category to which an issue belongs to. For example: Hardware is the category of all issues relevant to monitor or a computer mouse etcetera.
- Close issue: The final confirmation of resolution of the issue, done by the initiator, after the support role employee resolves the technical problem.
- Dart: Programming Language used in Flutter.
- Employee: An educated individual hired by ABL.
- Fix issue: When a support team member resolves an issue.
- Flutter: One of the front-end development tools used.
- GUI: Graphical User Interface.
- ILS: Issue Logging System.
- Initiator: Any employee that logs in an issue on the app.
- Issue: Refers to a technical issue.
- Location: City of the branch where the issue was logged in.
- Log an issue: Fill a form with the required information about the technical issue and upload it.
- Manager: Manager of a support team.

❑ Monitor: User that can review reports based on the number of issues logged through the application.

❑ SDS: Software Designs Specification

❑ Sub-category: The more specific category of an issue. For example: any issues related to keyboard will have the sub-category as keyboard.

❑ Sub-location: Area of the branch where the issue was logged in.

❑ Support: Any employee part of a support team.

❑ Support Team: The employees at ABL that can resolve issues in a specific category form the support team of that category.

❑ Technical Issue: A hardware or software problem related to electronic devices that have been issued to an employee or a branch by the bank and is property of the bank.

❑ UI: User Interface

❑ Work Category: the expertise area of a support team member.

1.5 Important URLs

API Hosting URL: https://ilslumsapp.000webhostapp.com

ILS' Repository on GitHub: https://github.com/maleehamasood/LUMS-CS360-Project

Trello Board Link: https://trello.com/invite/b/qHvWmR6s/8583d3dd72b59cf3729dd003a6b0a216/ils-cs360-project

Section 2 Summary of the Implementations

Kindly note that the enumeration of the functional requirements as per those mentioned in Section 1.2 of the Software Design Specification Document. Further, the use cases have been split into function requirements and so their mentioning is repetitive.

Functional Requirements	Implemented
1	Yes
2	Yes
3	Yes
4	Yes
5	Yes
6	Yes
7	Yes
8	Yes
9	Yes
10	Yes
11	Yes
12	Yes
13	Yes
14	Yes
15	Yes
16	Yes
17	Yes
18	Partially
19	Partially
20	Yes
21	Yes
22	Partially
23	Partially
24	Partially
25	Partially
26	Partially
27	Partially
28	Partially
29	Yes
30	Yes
31	Yes

Section 3 Rational Behind Incompleteness

After discussion from the course staff, both the instructor and the teaching assistants, we agreed to drop the iOS compatible version of the ILS application. While in the development phase we still had hopes that we could achieve this, however when it came to being practical and honest about it, only 2/5 members had access to a macOS device and one out of these two was not very familiar with the front end, since they were working on the backend. Thus, keeping the submissions other than the apk file in mind, we dropped the idea to pursue the construction of an iOS friendly ILS.

Functional Requirements 18 and 19: We have only made 3 of the 4 graphs in the requirements. The last one was growing extremely large due to the nature of its contents and was not easily manageable by us due to this. Its presence was also expectedly having an adverse effect on the production of the other graphs in terms of time because of its excessive data fetching.

Functional Requirements 22 to 28: All emails are being sent. However, while, flutter generally has lots of documentations available online, push notifications were something we felt very unsure about even after going through the relevant documentations and articles. We just were not able to clearly wrap our head around how to send notifications to idle devices and tackle scenarios of no Wi-Fi availability and such. Thus, we dropped this idea too.

Section 4 Overall Description

4.1 Effectiveness and breadth of the system

The first description that we think the application deserves that it is a product of the programming framework flutter. Each screen is a spiral of widget in widget in more widgets. Classes are each a separate dart file. All these can be found in the lib folder of the application. All related pictures and fonts are in the assets folder. The backend database used is ClearDB that is being accessed by a self-written API hosted on a free website www.000webhost.com.

As a whole application, ILS does achieve its purpose of easing and facilitating the usually dreaded task of requesting technical repairs. Already registered users can use the application on their Android devices to share their problems with technical staff. However, since it is a prototype and because of financial constraints, the scalability of the application is restricted to a database of size 5MB, with the server capable of handling about 10,000 requests per day.

4.2 Efficiency at the design and the architecture level

4.2.1 Efficiency at the Design Level

4.2.1.1 UI

To make ILS accessible and generalizable for use by many different users, the User Interface of the application has been designed to be kept simplistic minimal. It also has easy to read, to the point text and where necessary, images and icons to ease the task of navigation through the application.

4.2.1.2 Color Contrast

A stark contrasting color theme has been used throughout the application: Orange, Blue and White. This set of colors makes the application feel more vibrant and makes the text and icons more prominent, which are often colored white on non-white backgrounds and gray on white background.

4.2.1.3 Minimal Text

Because one of the design goals of the application was to make the process of logging issues swifter, thus, we have designed the application to contain minimal text to avoid the situation of having many wordy screens filled with long sentences.

4.2.1.4 Use of Pictorial References

To allow a user not to get 'lost' within the application, along with descriptions of buttons, guidance pictures have been added to improve understandability of the features available.

4.2.1.5 Navigation

Clear navigation instructions like arrow keys for transition in between screens and swiping fingers to guide the user that a screen needs to be swiped to view more have been added to ensure that all expected navigations are easy for a first-time user.

4.2.1.6 Email Notifications

To keep the relevant individuals in the loop about the status of any issue, email notifications will be sent to them so that users stay aware of the current processing of the technical issue without having to repeatedly log in to the application.

4.2.1.7 Full Sized Buttons and Minimal Text Fields

The users of this application will interact with it by using their fingers for tapping. Thus, to avoid unintentional clicks, we have added in complete sized buttons (after referring to the material design guidelines) and have reduced the number of places where input using the keyboard is required.

4.2.1.8 Information Screens

We understand that a new employee at ABL might be overwhelmed by the terminology being used in this app, thus, to facilitate such an employee's usage of the application, we have added in information screen that can be accessed from the navigation bar at the top of the screen. These information screens describe

the terminology used in the application to remove any confusions of what is expected.

4.2.1.9 Response Dialogue Boxes

To make ILS a responsive application, whenever a user encounters an error or successfully submits something, responses are shown so that the user is not left wondering if something happened or not.

4.2.1.10 Visuals

A requirement of our client was to generate reports that can help a monitor, to put in simple words, monitor the issues logged. To prevent bombarding this role and the Manager with lots of numbers, we decided to visually present this information in the form of graphs.

4.2.2 Efficiency in the Architectural Level

4.2.2.1 Use of Three Layer Architecture

The complete application can be broken down into three different layers of architecture: presentation, business and data access. The interface of the application is the presentation layer. The logic introduced (like navigation, outcomes of button presses) form the business layer. The backend and methods to access it (example: the API) form the data access layer. By separating all three layers out, modifying the application for further development and features has been made easier. It has also overall improved the ease of dealing with the code of the ILS.

4.2.2.2 Use of Flutter

To especially separate out the presentation and business layer, flutter has been a huge help. Flutter has in built widgets which are called to build screens and within these widgets, there are placeholders where logics can be implemented using methods.

4.2.2.3 No Local Database

The first architectural decision in order to improve the efficiency of the application is the scarce use of local memory, other than the size of the mobile application. We believe this adds to the efficiency of the application keeping in mind the fact that while employees at ABL will be our primary users, the truth still holds that many users in developing countries have access to low end mobile devices with finite memory available. Thus, rather than creating a local copy of the database on the device, our device functions with the use of a cloud database only.

4.2.2.4 Limited Bandwidth Usage

Building upon the idea that users of this application will be from a developing country, in an attempt to reduce the bandwidth usage of the application, we have designed the application to consist of minimal small sized images that are stored locally within the apk file of the application. The scenario of excessive bandwidth usage for rendering images online has been avoided.

While one might feel that this point contradicts the previous one (4.2.2.3), realize that the memory requirement of storing a complete database of logged in issues is much larger than storing a handful of images. Thus, the tradeoff between reducing bandwidth by storing images locally is more preferred in terms of our audience's view points as compared to reducing the memory usage of the application by a few bytes.

4.2.2.5 Security

Using the API, we have ensured the security of the mobile application as only employees with valid Employee ID and Passwords as well as an internet connection will be able to move on to the menu screen. Security has also been a requirement of our client and so, implementing the security factor within the architecture of the application was crucial.

4.2.2.6 Well Defined Boundaries of Responsibilities

Throughout the application, only the necessary data required for a screen is fetched from the database rather than fetching everything on login and then locally using the retrieved content for different screens. This is generally good practice. Further, in case of change to the database, freshly received data is shown to the user as compared to an older, cached version.

4.3 Visual Design/Aesthetics

The design of the application was a very tricky part to achieve for the reason that there is a vast difference between reading the guidelines available to follow and actually implementing and abiding by them. However, we ensured our aesthetics by sticking to the following practices:

- We followed a strict color scheme for theming the application.
- We developed a style sheet to ensure we do not deviate from the generalized text format.
- Where possible, codes of widgets like dialogue boxes, buttons and application bars were reused to ensure consistency.
- Every screen was checked and rechecked with the material design guidelines, especially dimensions of buttons.
- All our widgets are aligned as per the dimensions of the device in use to ensure that our application does not look misplaced on phones of various dimensions.
- We avoid the situation of having a wordy screen by not using too much text in the first place.
- We have ensured our icons, text and images are clearly visible and understandable.
- We do not face the issue of pixelated images.
- We have provided as many visuals to facilitate the user anywhere we felt it would help.
- We have used a contrasting color scheme matching the client's logo to ensure firstly, originality of application and its affiliation to ABL and secondly because the difference in the color set is very attracting.
- We have self-designed a logo for the application replacing the existing standard flutter application icon.
- We have used the swipe navigation technique to make the
- We have locked the orientation mode to portrait to prevent distorting the layout.
- Any feedback given to us by the reviewers has been taken very seriously.
- Any image used is wrapped around a container to ensure that it does not go out of dimensions.
- Our UI is responsive i.e. displays messages on actions by the user. This prevents a user from wondering whether an action happened or not.

4.4 Coding Guidelines Practices

4.4.1 Unique Files for Each Class

This has been a practice we followed to ensure that one file does not get overwhelmingly large.

4.4.2 Commenting

Flutter has functions that almost always only have widgets in them with their predefined properties. Thus, we barely felt the need to comment inside functions since most variables and properties were self-explanatory. For every function in general we have given the reason for the existence of function and what it does. Further, all essential rules of commenting like no useless comments, no confusing comments have been followed.

4.4.3 Naming Practices

For all names, be in variable, function or class, meaningful names have been decided on to prevent confusion for the reader.

Flutter itself has syntax rules that are recommended for coders like using lowerCamelCase for variables and functions that we have made sure to follow. Additionally, function names have been kept as verbs as per the usual practice. Classes names in flutter are nouns and in title case. This is also something we have made sure to follow. Each file has also been given the same name as the class it contains – the naming technique is the class name in small letters with _ in between words.

4.4.4 Global Variables

No global variables have been used that are accessible through more than 1 file. Any variable needed in a file is specially being fetched from the server again.

4.4.5 Indentation

The automatic formatter suitable for Dart code has been used for indentation.

4.4.6 Length of Functions

To avoid confusion, we have tried our best to make each function do only one major task. However, sometimes because of having numerous features of widgets, some methods have been intentionally kept long.

Section 5 Extra Work

As a group, all of us feel that we have stretched ourselves up until the very last day to put in effort into producing an application that reflects a whole semester's worth of our learning. As such we don't think we have done anything extra in terms of a new discovery or doing double the work that we previously signed up to do so. However, throughout this whole phase, we only had one aim in mind- make sure our application is easy and simple to use. We did not want to add any meaningless features in it. I think all of us went out of our way to achieve that. We do feel that we have achieved that at some level, albeit there are some flaws, and were hoping that the course staff would be able to appreciate that too.

Section 6 Contribution Statement

Group Member	Contribution	Hours
Aadam Nadeem	Application Graphics design. Application Logos and images design. UI layout design. Complete menu screens. Color Scheme and miscellaneous design. Menu screen layout and front-end. Helped other members in all other screens. Project data coordination.	100
Maleeha Masood	Login Screen Graphs Visuals Complete Documentation including Project Management Document GitHub Management Group Coordination	100
Malik Ali Hussain	Employee Roles/Issue status update screens front end and back end. Helped in all other screens partially. Helped in back end integration of all screens. Trello coordination. Graph screens back end integration partially.	100
Muhammad Raheem Zafar	Client Pitching. Group Coordination. Partial UI layout design. List generation front-end and partial backend. Updating Status of Issues front end and partial backend. Partial help in back end integration.	100
Shahrukh Kemall	Relational Database Design Database set-up on SQL ClearDB (Cloud Database) as Heroku add-on Backend php API development Backend php API Hosting Services.dart for php API calls Backend and Frontend integration via Services.dart Partial email notifications set-up	100

Following the matching stylesheet, commenting rules, the material design guidelines were the responsibility of the individual making the screen.

Note that while some boxes do look smaller than the others, we personally believe that everybody has put in the maximum effort that they could have into the application.

Section 7 Instructions for Application Use

7.1 Pre-requisites

a) Internet Connection
b) Company Provided Login Credentials
c) Installed application from its apk file

7.2 Permissions Required

a) Read and Write to Device
b) Internet Connectivity

7.3 Instructions

Upon opening the application, the first screen visible is the login screen, where the user is expected to enter their credentials. If the credentials are incorrect, then an error message is displayed. Otherwise the menu screen is displayed.

Now, there five different menu screens one for each role. We will move forward with the Manager screen since it has the largest subset of features. Only some of the Admin features are not subset of the features of the Manager Screen but they are self-explanatory and can be used without any instructions.

On the manager screen, screen 4 buttons are visible. Press the log an issue button, to enter in a new issue. Fill in the required information to finally submit the form. Click on the 'i' in the application bar to find details about the screen. Go back by pressing the back key in the application bar.

Now on the menu screen, press Check all issues to view the issue logged in your city. You can click on any issue to view more details about it or to take it up. Issues taken up will then be visible in the 'Issues taken up' button in the 'More' menu. Click on the 'More' menu, then click on the 'Issues taken up' button to view the issues taken up by you. Click on any issue to view the details of an issue. Click on the dropdown menu to get options of updating the status. Chose an option accordingly and click on the update button to confirm the change.

Come back to the More menu by pressing the back button. To view Issues taken up by your team click on the 'Team issues' button, here you can view the issues that are taken up by your team. By clicking on any issue you will be able to see the details of an issue and a button to 'Re-assign' clicking this button will lead you to a screen which shows a list of employees to which the issue can be re-assigned. By clicking on an employee, the issue will be reassigned to that employee.

The 'monitor reports' button will lead you to a screen which can be scrolled horizontally to view graphs with different categories and constraints.
Finally, the Logout button in the More functionality will lead you to the login screen and log you out of the application.

To close the application, just go back from your android on screen buttons.