



NITK

SURATHKAL

SOFTWARE ENGINEERING LAB (CS303)



MINI PROJECT

(RESTAURANT MANAGEMENT APP)



RESTAURANT MANAGEMENT APP

TEAM MEMBERS

KORADA SRINIVAS KALYAN – 191CS130



YENUMALA VENKAT KUMAR – 191CS263



L S V SANDEEP M – 191CS226



AJAY TUMMALA – 191CS259



- **Certificate**
- **Acknowledgement**
- **Executive Summary**
 - Description
 - Objectives
- **Project Management**
 - Introduction
 - Process Model
 - Risk Management
 - Project Plan
 - Functional point analysis
 - Effort Estimation
- **Requirement Gathering and Analysis**
 - Introduction
 - Data Flow Diagram
 - ER Diagram
- **Design**
 - Introduction
 - Data Design
 - Architecture Design
 - Interface Design
 - Backend Code Overview

- **SOFTWARE TESTING**

- **Introduction**

- **Why Software Testing is important**

- **Why is testing necessary**

- **FIREBASE Test Lab**

- **Get started testing for Android with Firebase Test Lab**

- **Step 1: Prepare your test for uploading to Test Lab**
 - **Step 2: Choose your testing device**
 - **Step 3: Review test results**

- **Get started with Robo tests**

- **Analyze Firebase Test Lab Results**

- **Interpret test history results**

- **Interpret test matrix results**

- **Interpret Robo test results**

- **Interpret results from a single test execution**

- **Performance metrics**

- **Bugs Fixed**

- **Security under consideration**

- **MAINTENANCE**

- **INTRODUCTION**

- **NATURE OF MAINTENANCE**

- **NEED OF MAINTENANCE**

- **MAJORITY OF MAINTENANCE COSTS**

- **CATEGORIES OF MAINTENANCE**

- **Corrective Software Management**
- **Adaptive Software Maintenance**
- **Perfective Software Maintenance**
- **Preventive Software Maintenance**

- **KEY ISSUES IN SOFTWARE MAINTENANCE**

- **Technical issues**
- **Management issues**
- **Cost estimation and measures**

- **MAINTENANCE PROCESS**

- **MAINTENANCE PLANNING ACTIVITY**

CERTIFICATE

This is to certify that the project report entitled "**SWILLER – DINING MADE SIMPLE**" done by Korada Srinivas Kalyan, Yenumala Venkat Kumar, L S V Sandeep M, Ajay Tummala is an authentic work carried out by them.

It embodies the work done by them during semester V of their course Software Engineering Lab(SE303) under the due supervision of **Anjali Jose**, Annappa B

Date: 30th October

Signature of the Guide

Name of the guide

ACKNOWLEDGMENT

We are indebted to **Anjali Jose** for her guidance and patience. It is her guidance that led us to envision our project " SWILLER – DINING MADE SIMPLE " to a full-fledged solution for Restaurant Management System.

We are also thankful to Annappa B for his guidance throughout the project.

We again thank all our teachers and all the people who helped in creation of this project.

EXECUTIVE SUMMARY

DESCRIPTION

The main aim of this project –**RESTAURANT MANAGEMENT APP** is to develop the software application for the process of booking tables and ordering food prior to their visit to the restaurant or they can order their food even if they are inside the restaurant which will avoid waiting at a restaurant.

We all know how important time is, waiting at a restaurant takes more time than expected. This application can be used by employees in a restaurant to handle the customers, their orders and can help them easily find free tables or place orders. The services that are provided is food ordering and reservation table management by the customer through the system online.

People nowadays hardly want to move from their places for shopping and not even for eating this is because of their hectic schedule. In such circumstances, we have decided to create a system for restaurants. This system is well known as Restaurant Management Services. This will help the restaurant owner in many ways such as ordering goods, inventory control, generating bills, managing tables, managing menus, and various customer services.

For example, significantly the number of patrons for a particular time interval, the manager can easily decide whether more employees and chefs are required or more inventory required.

OBJECTIVES

- Our primary objective is to automate the managing system of a restaurant for easy use of both client and the management.
- To provide fast, efficient and reliable system that manages records of the transactions
- To provide an efficient system that allows clients to order food easily
- Effective maintenance of the staff
- To build a feature rich for automatic billing
- To provide a platform itself being a select and order the menus and book services
- One of the main objectives is to maximize profit by increasing efficiency and decreasing overheads without compromising customer satisfaction

PROJECT MANAGEMENT

INTRODUCTION

Project management uses a systematic and disciplined approach to develop software. It consists of all the umbrella activities.

It includes the following activities:

- Estimation
- Project Scheduling
- Risk Management
- Quality Management
- Change Management

Project management involves the planning, monitoring and control of the people, process and events that occur as software evolves from a preliminary concept to an operational implementation. Effective software project management focuses on the four principles: people, product, process and project.

The People

Software engineering institute has developed a people management capability maturity model (PM-CMM). The people management maturity model defines the key practice areas for software people like: recruiting, selection, performance management, training, compensation, career development, organization and work design and team/culture development.

The Product

Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered and technical and management constraints should be identified. Scope identifies the primary data, functions and behaviors that characterize the product.

The Process

A software process provides the framework from which a comprehensive plan for software development can be established. Framework activities are populated with tasks, milestones, work products and quality assurance points. These activities characterize the software product and the project team.

Umbrella activities i.e. software quality assurance, software configuration management and measurement overlay the process model.

LIFE CYCLE MODELS



Software Development Life Cycle Models and Methodologies

Software development life cycle (SDLC) is a series of phases that provide a common understanding of the software building process. How the software will be realized and developed from the business understanding and requirements elicitation phase to convert these business ideas and requirements into functions and features until its usage and operation to achieve the business needs. The good software engineer should have enough knowledge on how to choose the SDLC model based on the project context and the business requirements.

Therefore, it may be required to choose the right SDLC model according to the specific concerns and requirements of the project to ensure its success.

Types of Software developing life cycles (SDLC)

- Waterfall Model
- V-Shaped Model
- Evolutionary Prototyping Model
- Spiral Method (SDM)
- Iterative and Incremental
- Agile development

MODEL USED IN OUR APPLICATION IS ---- WATERFALL MODEL

PROCESS MODEL

Waterfall process model is used in our project **SWILLER – DINING MADE SIMPLE**.

This model is generally applied to the projects where the requirements are very clear and the customer keeps on patience.

Different phases of model are as follows:

1. Software Requirements Analysis

In this, software engineer understand the nature of a program to be built, he must understand the information domain for the software as well as required function, behavior, performance and interface. Requirements for both the system and software are documented and reviewed with the customer.

2. Design

It has four distinct attributes of a program: data structure, software architecture, interface representations and procedural details. It is documental and becomes part of software.

3. Code Generation

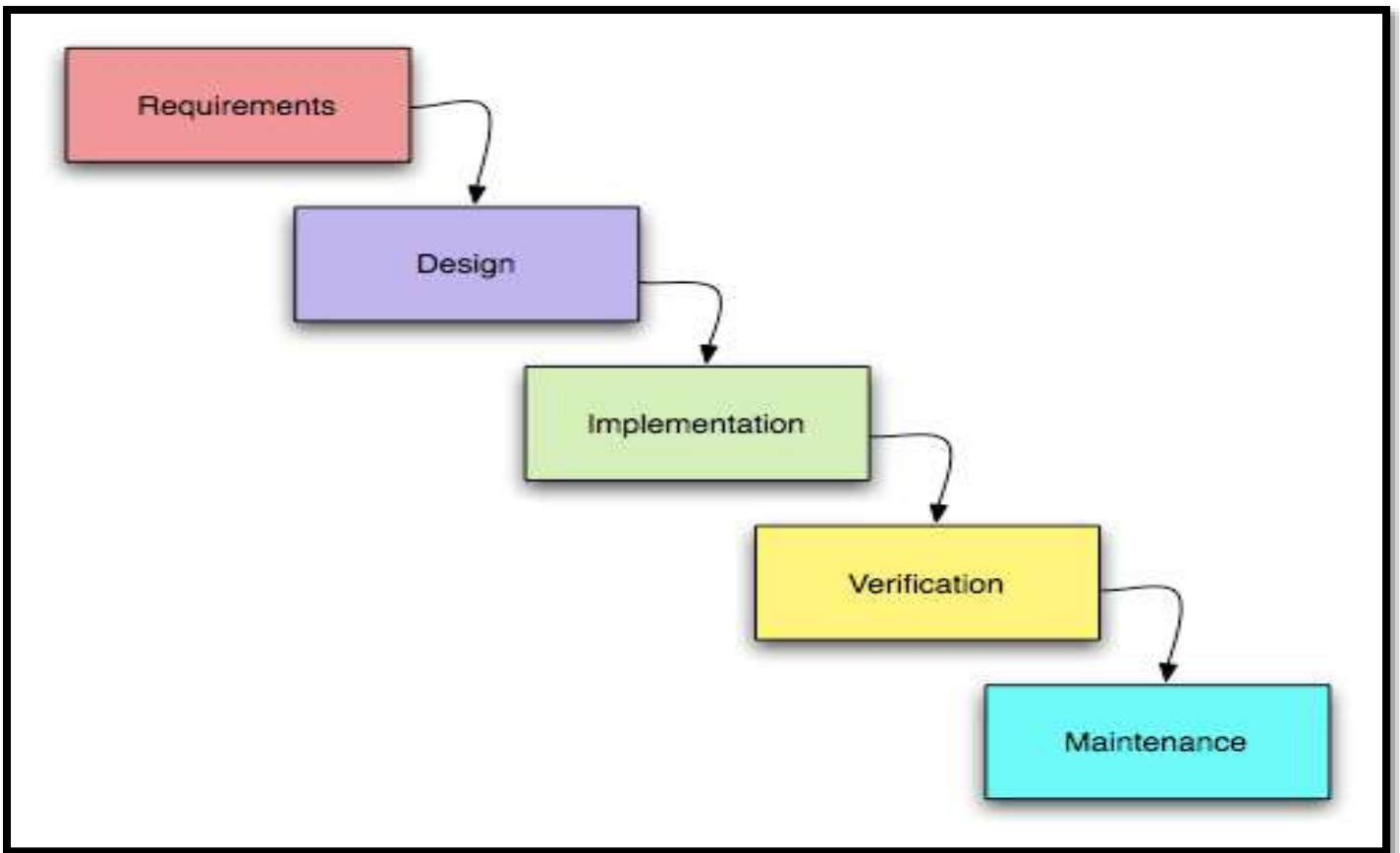
Design must be translated into a machine readable form which is done by code generation.

4. Testing

It focuses on the logical internals of the software, ensuring that all the statements have been tested, and on the functional externals, that is conducting test to uncover errors and ensure that defined input will produce actual results.

5. Support

This is a phase when software will undoubtedly undergo change after it is delivered to the customer. Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment, or because the customer requires functional or performance enhancements. Software support/maintenance reapplies each of the preceding phases to an existing program rather than a new one.



Advantages of waterfall model

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle. High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

Why did we use the waterfall model?

- We have used this model because the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements.
- Ample resources with required expertise are available freely
- The project is short (wasn't that short anyway).

RISK MANAGEMENT



Risk Analysis and Management are a series of steps that help a software team to understand and manage uncertainty. Many problems can plague a software project.

A risk is a potential problem- it might happen or it might not. But regardless of the outcome, it's a really good idea to identify it, assess its probability of occurrence, estimate its impact, and establish a contingency plan should the problem actually occur.

Software is a difficult undertaking. Lots of things can go wrong. and frankly many often do. It's for this reason that being prepared- understanding the risks and taking proactive measures to avoid or manage them- is a key element of good software project management.

Recognizing what can go wrong is the first step, called "Risk Identification". Next, each risk is analyzed to determine the likelihood that it will occur and the damage that it will do if it does occur. Once this information is established, risks are ranked, by probability and impact. Finally, a plan is developed to manage those risks with high probability and high impact. The work product is "Risk Mitigation, Monitoring and

Management (RMMM) Plan" or a set of risk information sheets is produced.

RISKS	%	RESULT	RMMM
The team may lose all the project artifacts any time during the project and thus will be unable to deliver the application to the customer. Such an unlikely event may be caused by a hard disk being wiped out by a virus, hard disk failure, etc.	4%	Catastrophic	<p>Mitigation Plan : Create backups of Project on regular intervals to cloud and make sure every documentation is up to date.</p> <p>Contingency Plan: Talk to customer for project extension and if he agrees then start the coding part again on the basis of documentation created earlier</p>
Customer requirements might change, since our software and system is made in a linear fashion, changing of requirements can be a big problem.	20%	Critical	<p>Mitigation Plan: create SRS properly during phase communication</p> <p>Contingency Plan: Try to convince him to accept ongoing project else ask for project extension</p>

You may not have enough human resource to finish the project at the deadline	50%	Critical	Mitigation Plan: Select the talented and experienced member to join the Project Team Contingency Plan: Talk to client for project extension and apologize to him
This website may lack security features	10%	Medium	Mitigation Plan: You can request the development team to check and add these functions to the website Contingency Plan : Negotiate with the client

PROJECT PLAN

Software project scheduling is an activity that distributes estimated effort across the planned project by allocating the effort to a specific software engineering tasks.

When you develop a schedule, compartmentalize the work, represent the task interdependencies, allocate effort and time to each task, define responsibilities for the work to be done, and define outcomes and milestones.

In order to build a complex system, many software engineering tasks occur in parallel and result of work performed during one task may have a profound effect on work to be conducted in another task.

These interdependencies are very difficult to understand without a schedule. It's also virtually impossible to progress on a moderate or large software project without a detailed schedule.

ASSUMPTIONS:

1. Authentication – Swiller has 2 types of login facility

Admin – Each admin will have their own login credentials and can login according and can have a view on the progress of the hotel and can make changes accordingly.

Users – Users can access the app by login with their credentials. A new user could Sign up with unique email and can access the app anytime.

2. Admin

Dashboard – Admin dashboard consists of the information regarding the individual and the details regarding the management.

Menu updates – Admin can check the menu and can make changes accordingly.

Staff List – Similarly, admin can know the details of the staff and will be able to manage them.

3. Payments

Bill generations – Every time when a client makes some purchase, the bill will be generated automatically and stores in the database and add to the aggregate value.

4. Table Reservation

Availability – Each user check the current status of the tables and can book them accordingly.

Booking Status – Once the table is chosen, the booking status will be shown to the customer

5. Ordering

Takeaway – Customers can order food online and can takeaway at the time of arrival without delay.

Dine – When the customer likes to have food in the restaurant, they can order food through the application without need of waiting for the waiter.

6. Menu

Daily Updated menu – The menu gets updated by the management whenever needed based on their requirements.

7. About –

This portal consists of history regarding the hotel along with some customer feedbacks.

Project Team :

The type of project team we have perfectly fits to the sub category called Self-Managed Teams.

Typically, members of self-managed teams are employees of the same organization who work together, and even though they have a wide array of objectives, their aim is to reach a common goal. There is no manager nor authority figure, so it is up to members to determine rules and expectations, to solve a problem when it arises, and to bare shared responsibility for the results.



REQUIREMENT GATHERING AND ANALYSIS

INTRODUCTION

Requirements Process is the sequence of activities that need to be performed in the requirements phase and it culminates in producing a high quality document containing the software requirements specification (SRS).

The requirement process consists of three basic tasks:

- Problem or Requirements Analysis
- Requirements Specifications
- Requirements Validation



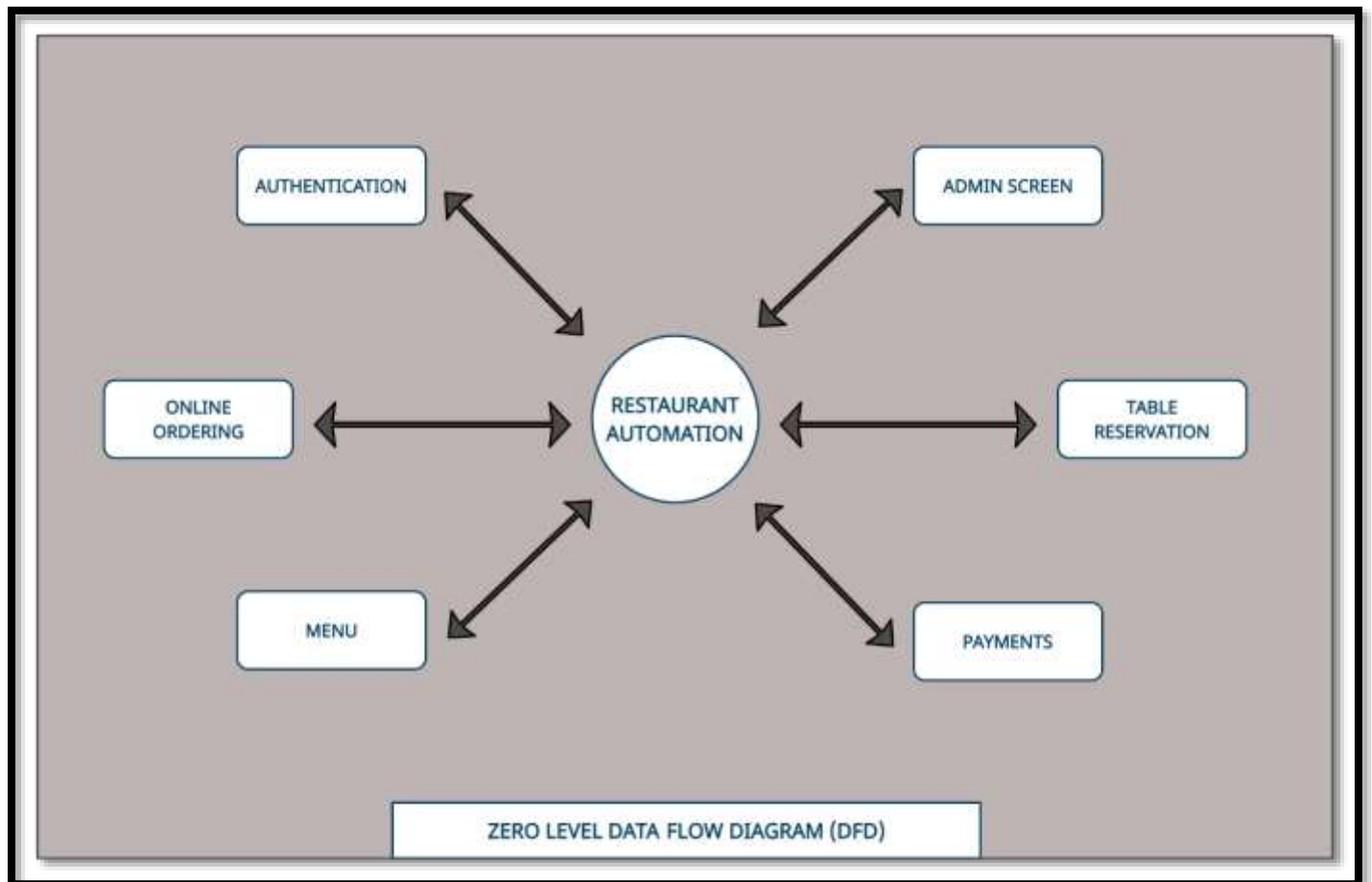
Problem Analysis starts with a high-level problem statement. During analysis the problem domain and the environment are modeled in an effort to understand the system behavior, constraints on the system, its inputs and outputs etc. The basic purpose of this activity is to obtain a thorough understanding of what the software needs to provide

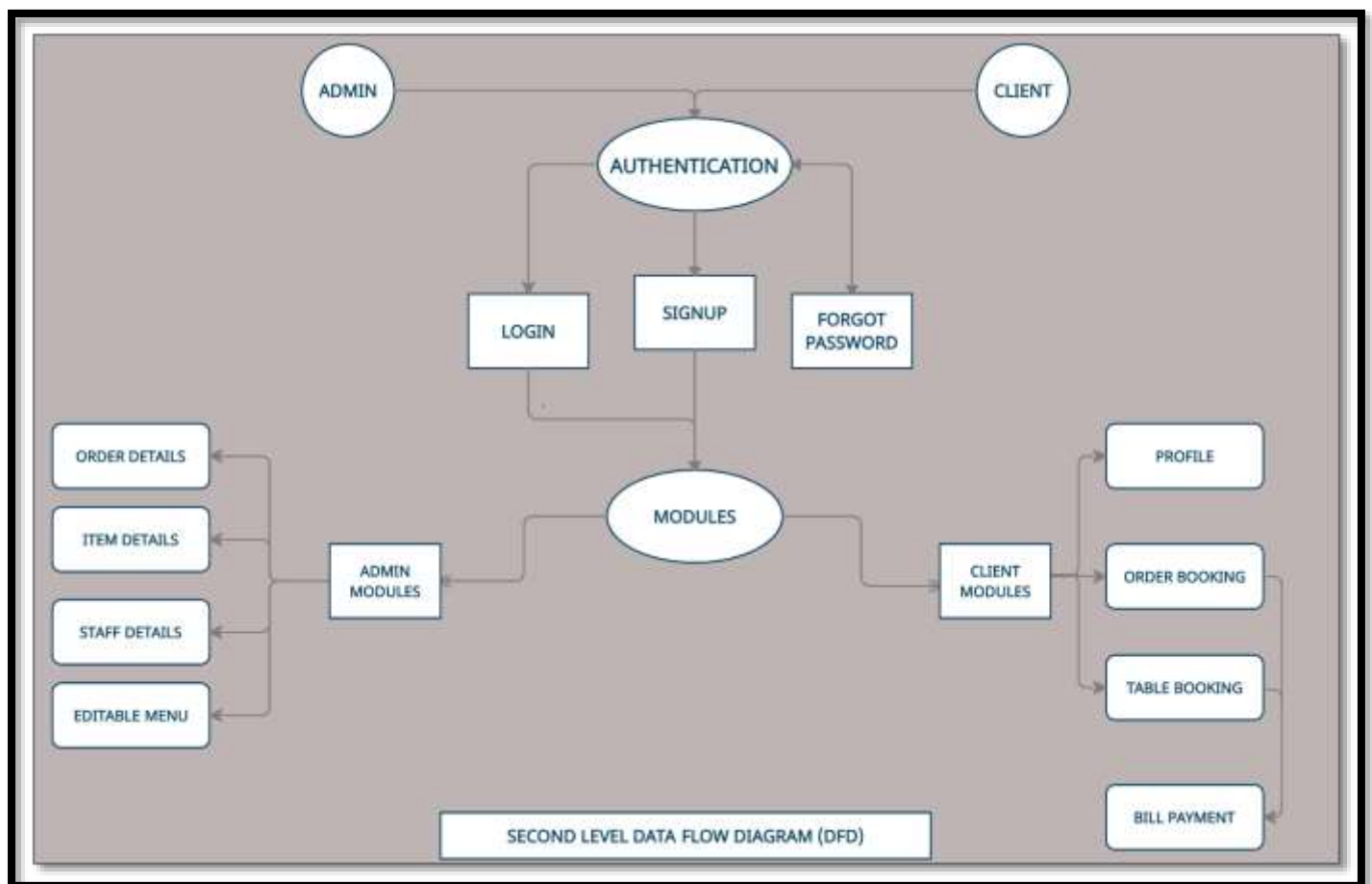
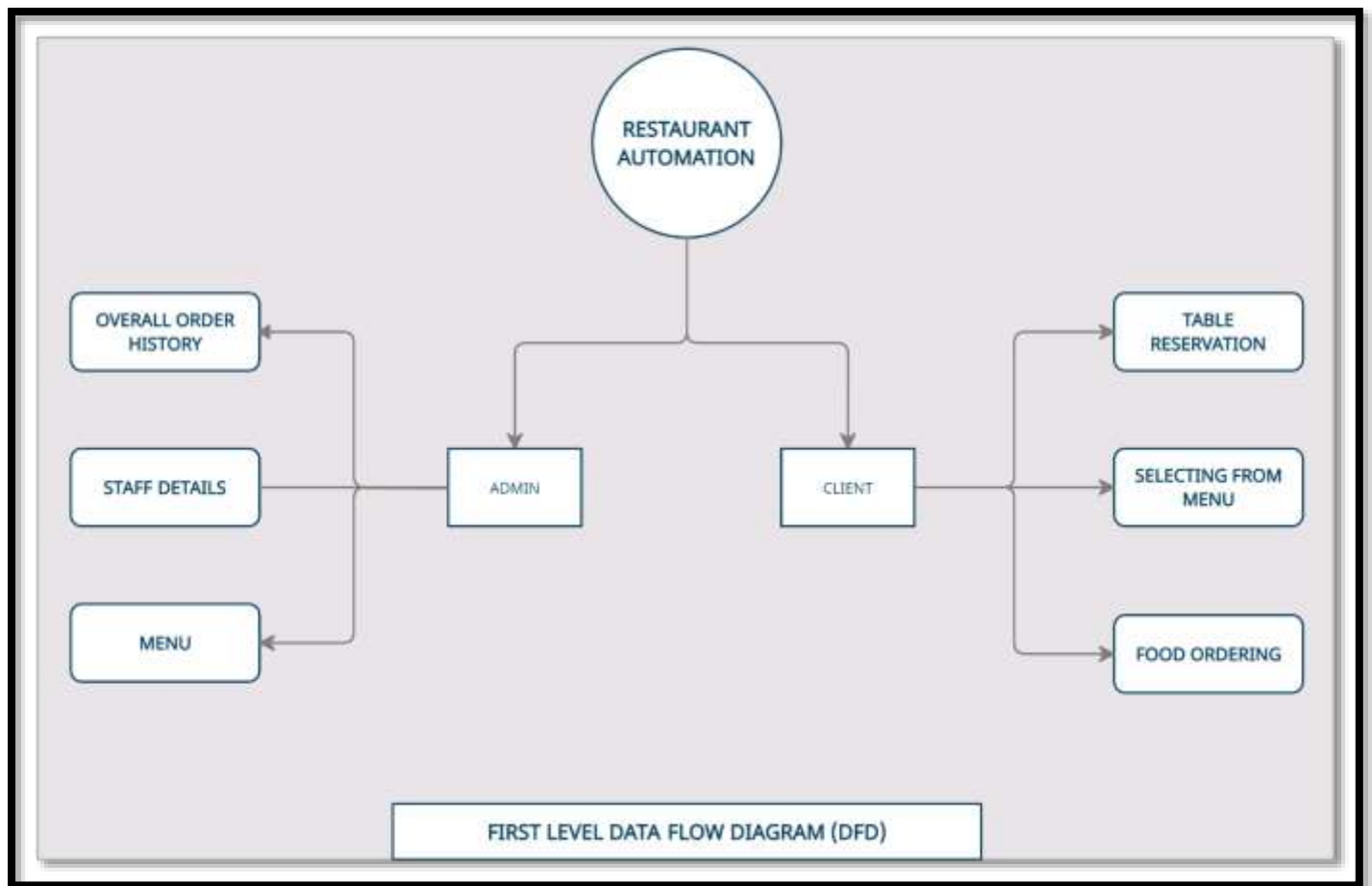
Requirements Specification focuses on clearly specifying the requirements in a document. Issues such as representation, specification languages and tools are addressed during this activity. As analysis produces large amounts of information and knowledge with possible redundancies, properly organizing and describing the requirements is an important goal of this activity.

Requirements Validation focuses on ensuring that what has been specified in the SRS are indeed all the requirements of the software and making sure that the SRS is of good quality. The requirements process terminates with the production of the validated SRS

DATA FLOW DIGRAMS

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated





DATA DICTIONARY

The data dictionary provides an organized approach for representing the characteristics of each data object and control item. It has been proposed for describing the content of objects defined during structured analysis. A Data Dictionary is very important in the software development process because of the following reasons:

- A Data Dictionary lists standard terminology for use by an engineer working on a project.
- The Dictionary provides the analyst with means to determine the definition of different data structures in terms of their component elements.

The format of Data Dictionary includes the following information

- Name-the primary name of the data or control item, the data store or an external entity.
- Alias-other names used for first entity.
- Description-a notion for representing content
- Type-type of the data.

Field Name	Type	Description
Username	String	User/Admin Login email
Password	String	User/Admin Login password
Staff_name	String	Staff name
Staff_num	Int	Staff contact number
Staff_img	String	Staff image path
Item_name	String	Name of the dish
Item_price	Int	Cost of the dish
Item_img	String	Item image path
Profile_name	String	User name
User_img	String	User image path
User_Phone	Int	User's contact number
User_city	String	User's city
User_country	String	User's country
Table_num	Int	Table number
Slot_num	Int	Slot number

DESIGN

INTRODUCTION

The design activity begins when the requirements document for the software to be developed is available and the architecture has been designed. During design we further the architecture.

Software design is a process of problem solving and planning for a software solution. After the purpose and specifications of software are determined, software developers will design or employ designers to develop a plan for a solution. It includes. Low-level component and algorithm implementation issues as well as the architectural view. The design of a system is a blueprint or a plan for a solution for the system. Here we consider a system to be a set of modules with clearly defined behavior, which interact with each other in a defined manner to produce some behavior or services for its environment.

A design should clearly be verifiable, complete (implements all the specifications), and traceable (all design elements can be traced to some requirements). However, the two most important properties that concern designers are efficiency and simplicity, Efficiency of any system is concerned with proper use of scarce resources by the system. Simplicity is perhaps the most important quality criteria for software systems.

Maintenance of the software is quite expensive. The simpler the software, the more easily it can be maintained.

The design activity mainly focuses on the following major areas of concern

COMPONENT LEVEL DESIGN: It establishes the algorithmic detail required manipulating the data structures, effect communication between software components via their interfaces, and implement the processing algorithms allocated to each component i.e. it transforms structural elements of software architecture into a procedural description of software components.

INTERFACE DESIGN: It deals with the process of developing a method for two or more modules in a system to connect and communicate. It describes how the software communicates with itself, with systems that interoperate with it, and with the users who use it.

ARCHITECTURAL DESIGN: It defines the relationship among the major structural elements. Here the main objective is to develop a modular structure and represent the control relationship between the modules

DATA DESIGN: It is the first and most important Design activity. It transforms the information domain model created during analysis into the data structures that will be required to implement the software. Hence, Data Design focuses on the definition of data structures

DATA DESIGN

Authentication

Field	Type	Null	Key	Default
User_Id	Text	NO	Primary	NULL
Timestamp	Date	NO		NULL
Password	Text	NO		NULL
Email	Email	NO		NULL

Users

Field	Type	Null	Key	Default
User_Id	Text	NO	Primary	Not NULL
User Img	Text	Yes		NULL
Phone	Int	Yes		NULL
Name	Text	Yes		NULL
Email	Email	NO		Not NULL
Country	Text	Yes		NULL
City	Text	Yes		NULL

Tables

Field	Type	Null	Key	Default
Table_Id	Text	NO	Primary	NULL
Is_Booked_1	Boolean	NO		False
Is_Booked_2	Boolean	NO		False
Is_Booked_3	Boolean	NO		False

Table Bookng

Field	Type	Null	Key	Default
Id	Text	NO	Primary	NULL
Slot_Num	Int	Yes		NULL
Charges	Int	Yes		NULL
Post_Time	Date	NO		NULL
Table_Num	Int	Yes		NULL
User_Id	Text	NO	Foreign	NULL

Staff

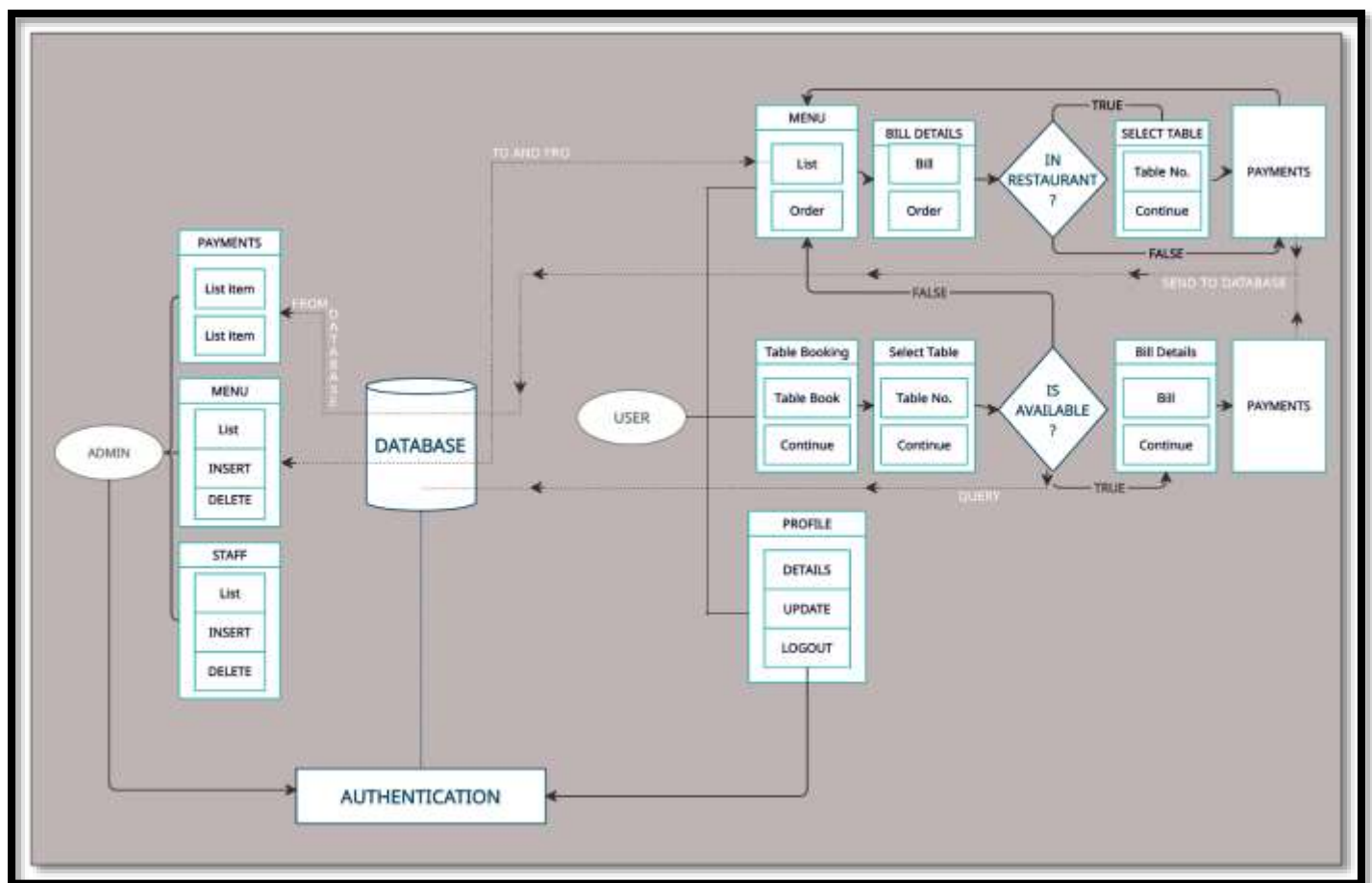
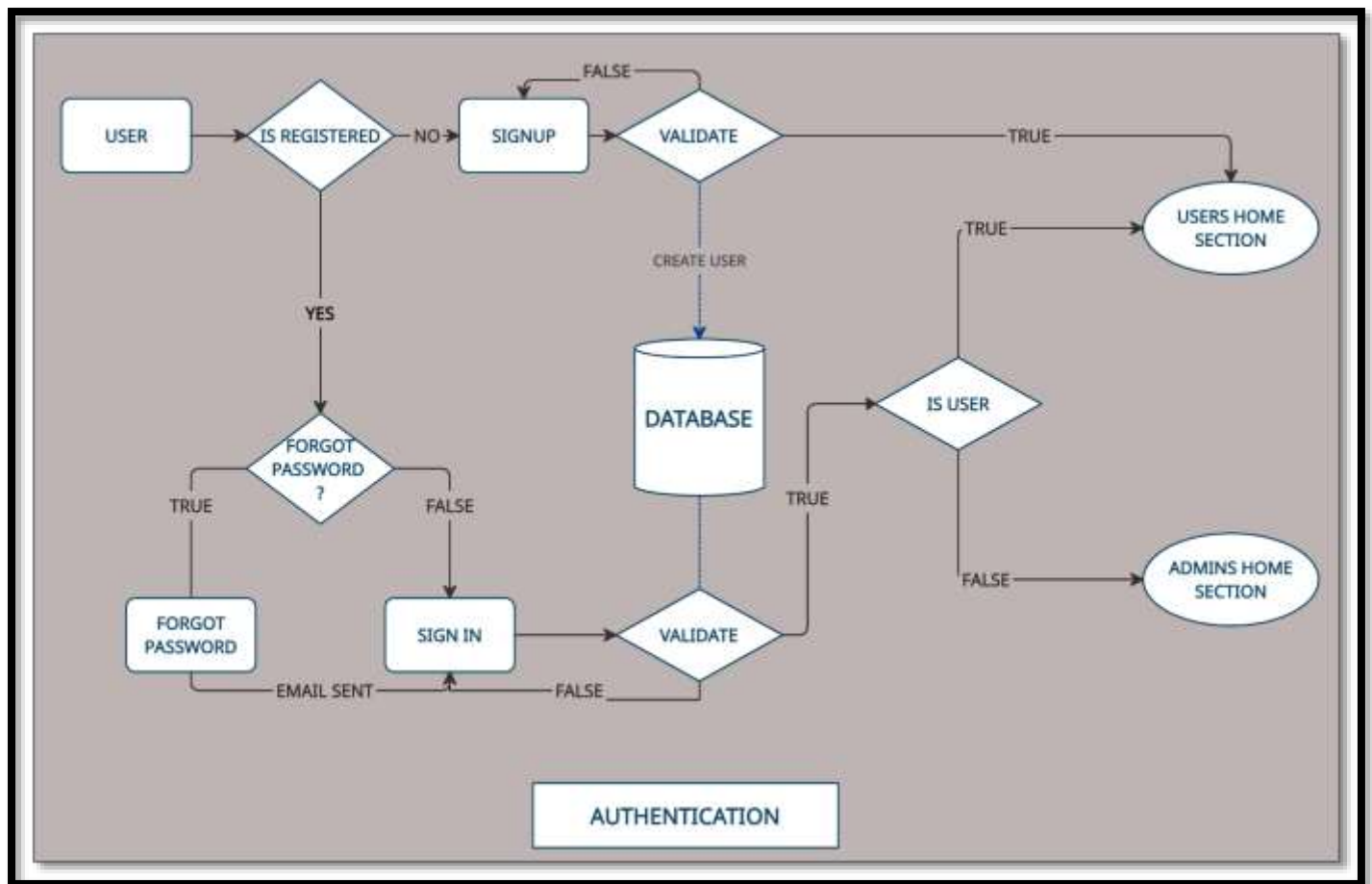
Field	Type	Null	Key	Default
Id	Text	NO	Primary	NULL
Name	Text	Yes		NULL
Phone	Int	Yes		NULL
Image	Text	Yes		NULL
Timestamp	Date	NO		NULL
User_Id	Text	NO	Foreign	NULL

Orders

Field	Type	Null	Key	Default
Id	Text	NO	Primary	NULL
Count	Int	Yes		NULL
Delivery_Charges	Int	Yes		NULL
Final_Total	Int	Yes		NULL
Items i. Count ii. Name iii. Price	Int Text Int	Yes Yes Yes		NULL
Paid	Boolean	Yes	Foreign	NULL
Timestamp	Date	NO		
Total	Int	Yes		
User_Id	Text	NO		

Staff

Field	Type	Null	Key	Default
Id	Text	NO	Primary	NULL
Name	Text	Yes		NULL
Timestamp	Date	Yes		NULL
Price	Int	Yes		NULL
Image	Text	Yes		NULL



SYSTEM ARCHITECTURE

SOFTWARE TESTING

INTRODUCTION

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Why Software Testing is Important

1. Helps in saving money
2. Security
4. Satisfaction of the customer
5. Enhancing the development process
6. Easy while adding new features
7. Determining the performance of the software

Why is Testing Necessary

Human errors can cause a defect or failure at any stage of the software development life cycle. The results are classified as trivial or catastrophic, depending on the consequences of the error.

The requirement of rigorous testing and their associated documentation during the software development life cycle arises because of the below reasons:

- To identify defects
- To reduce flaws in the component or system
- Increase the overall quality of the system

There can also be a requirement to perform software testing to comply with legal requirements or industry-specific standards. These standards and rules can specify what kind of techniques should we use for product development. For example, the motor, avionics, medical, and pharmaceutical industries, etc., all have standards covering the testing of the product.

The points below shows the significance of testing for a reliable and easy to use software product:

- The testing is important since it discovers defects/bugs before the delivery to the client, which guarantees the quality of the software.
- It makes the software more reliable and easy to use.
- Thoroughly tested software ensures reliable and high-performance software operation.

For example, assume you are using a Net Banking application to transfer the amount to your friend's account. So, you initiate the transaction, get a successful transaction message, and the amount also deducts from your account. However, your friend confirms that his/her account has not received any credits yet. Likewise, your account is also not reflecting the reversed transaction. This will surely make you upset and leave you as an unsatisfied customer.

Now, the question arises, why did it happen? It is because of the improper testing of the net banking application before release. Thorough testing of the website for all possible user operations would lead to early identification of this problem. Therefore, one can fix it before releasing it to the public for a smoother experience.

Firebase Test Lab

Test your app on devices hosted in a Google data center.

Firebase Test Lab is a cloud-based app testing infrastructure that lets you test your app on a range of devices and configurations, so you can get a better idea of how it'll perform in the hands of live users.

Get started testing for Android with Firebase Test Lab

Firebase Test Lab lets you test your app on a range of devices and configurations. This Get Started guide provides an implementation path for you to follow, as well as an introduction to Test Lab's Android offerings.

Key concepts

When you run a test or a set of test cases against devices and configurations you've selected, Test Lab runs the test against your app in a batch, then displays the results as a test matrix.

Devices × Test Executions = Test Matrix

Device

A physical or virtual device (Android only) you run a test on, such as a phone, tablet, or wearable device. Devices in a test matrix are identified by device model, OS version, screen orientation, and locale (also known as geography and language settings).

Test, test execution

A test (or a set of test cases) to be run on a device. You can run one test per device, or optionally shard the test and run its test cases on different devices.

Test matrix

Contains the statuses and test results for your test executions. If any test execution in a matrix fails, the whole matrix fails.

Step 1: Prepare your test for uploading to Test Lab

Available test types

You can run the following tests with Test Lab. Note that all test types are limited to running 45 minutes on physical devices and 60 minutes on virtual devices. Any uncaught exception will cause a test failure.

- **Instrumentation test or instrumented unit test:** A test you've written using the Espresso or UI Automator 2.0 frameworks. With this test, you can make explicit assertions about the state of your app to verify correct functionality using `AndroidJUnitRunnerAPIs`.
 - Visit [Run an instrumentation test](#) for instructions on how to prepare your test to run in Test Lab.
 - Refer to the [Android Developers documentation](#) for instructions on how to build an instrumentation test.
- **Robo test:** An automated test that analyzes your app's UI and then explores it methodically by simulating user activities, without requiring you to write any code. Visit [About Robo tests](#) for more information.
- **Game Loop test:** A test that uses a "demo mode" to simulate player actions in gaming apps. This is a fast and scalable way to verify that your game performs well for users. When you choose to run a Game Loop test, you can:
 - Write tests native to your game engine
 - Avoid writing the same code for different UIs or testing frameworks
 - Optionally create multiple loops to run in a single test execution (visit [About Game Loop tests](#) to learn more). You can also organize loops by using labels so you can keep track of them and re-run specific loops.

Tools to run your test

You can choose the following tools to run your test with:

- **Recommended for first-time users:** The Firebase console lets you upload an app and initiate testing from your web browser. See [Test with the Firebase console](#) for instructions on running tests using this tool.

- Android Studio integration lets you test your app without leaving your development environment. See [Test with Android Studio](#) for instructions on running tests using this tool.
- The gcloud command line interface enables you to run tests from the command line interactively, and is also well suited for scripting as part of your automated build and testing process. See [Test with the gcloud CLI](#) for instructions on running tests using this tool.

Step 2: Choose your testing device

Test Lab supports testing on several makes and models of Android devices installed and running in a Google data center. Testing on devices in Test Lab help you detect issues that might not occur when testing your app using emulators in Android Studio.

Step 3: Review test results

Regardless of how you initiate your tests, all your test results are managed by Test Lab and can be viewed online.

The **test result summary** is automatically stored and can be viewed in the Firebase console. It contains the most relevant data for your test, including test case-specific videos, screenshots, the number of tests that passed, failed, or got flaky results, and more.

The raw test results contain test logs and app failure details, and is automatically stored in a Google Cloud bucket. If you specify a bucket, you are responsible for the cost of the storage. If you don't specify a bucket, Test Lab creates one for you for free.

When you initiate a test from Android Studio, you can also review test results from inside your development environment.

Get started with Robo tests

Robo test is a testing tool that is integrated with Firebase Test Lab. Robo test analyzes the structure of your app's UI and then explores it methodically, automatically simulating user activities. Unlike the UI/Application Exerciser Monkey test, Robo test always simulates the same user activities in the same order when you use it to test an app on a specific device configuration with the same settings. This lets you use Robo test to validate bug fixes and test for regressions in a way that isn't possible when testing with the UI/Application Exerciser Monkey test.

Robo test captures log files, saves a series of annotated screenshots, and then creates a video from those screenshots to show you the simulated user operations that it performed. These logs, screenshots, and videos can help you to determine the root cause if your app crashes, and can also help you to find issues with your app's UI.



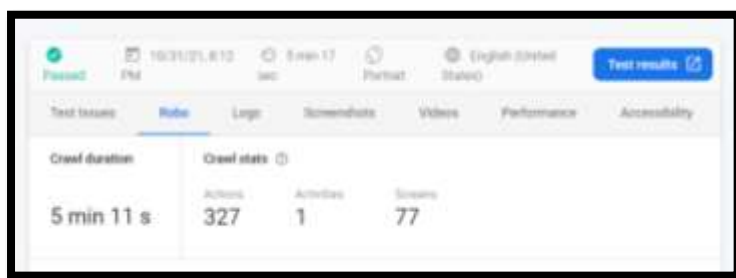
Robo test crawl stats

To help you interpret your Robo test results, Robo records stats during each test crawl. Test Lab displays the stats at the top of the Robo tab in your test's results page:

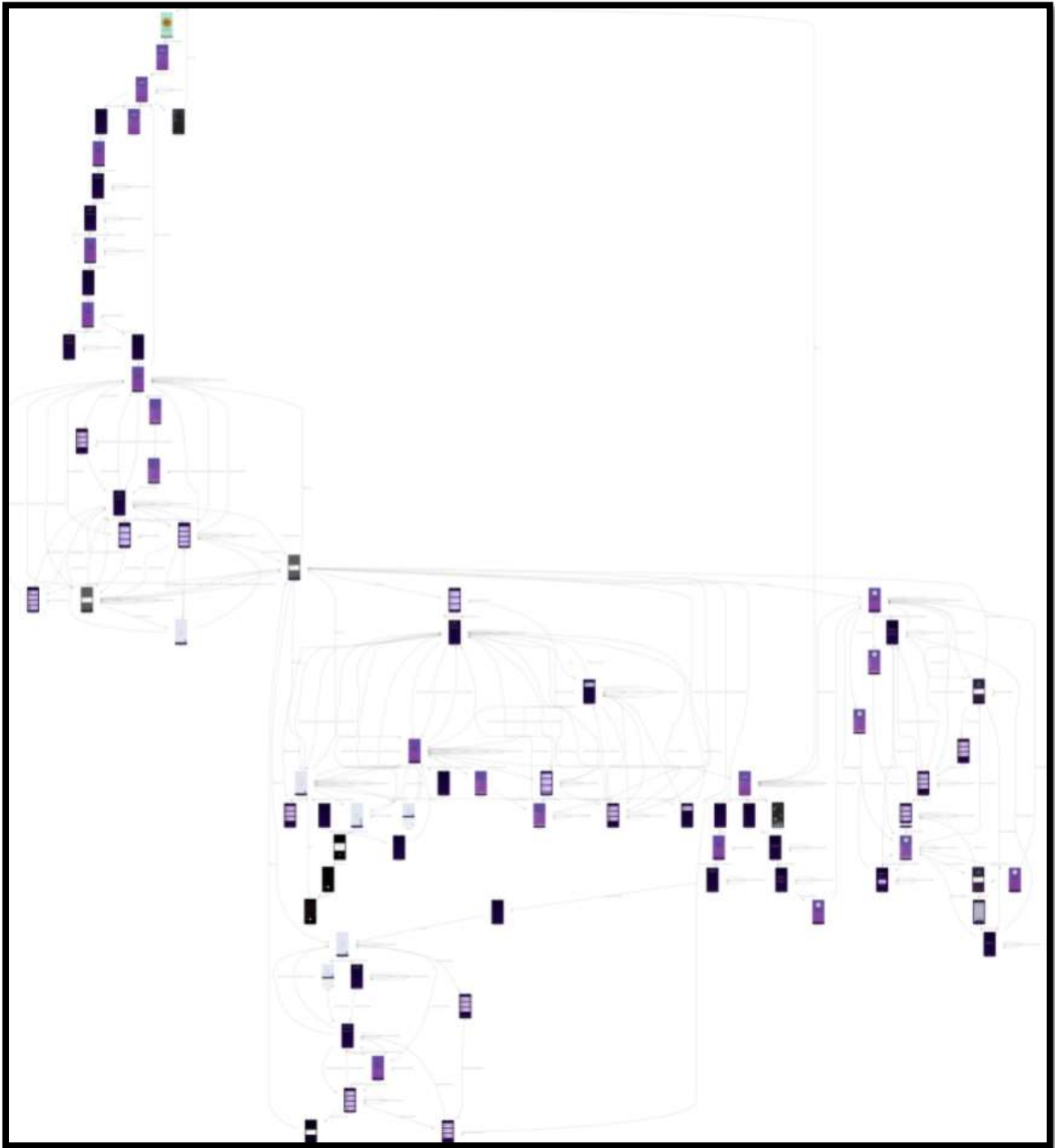
Actions: The total number of actions performed during the crawl, including Robo script actions, monkey actions, and Robo directives.

Activities: The number of distinct activities covered during the crawl.

Screens: The number of distinct screens visited during the crawl.



Test Lab also uses the stats to create a visual representation of the Robo crawl in the form of a crawl graph. The graph has screens as its nodes and actions as edges. By following the edges between screens, you can get an idea of how Robo traversed your app throughout the crawl.



Robo test timeout

Depending on the complexity of your app's UI, Robo test might take five minutes or more to complete a thorough set of UI interactions. We recommend setting the test timeout to at least 120 seconds (2 minutes) for most apps, and 300 seconds (5 minutes) for moderately complex apps. The default value for timeout is 300 seconds (5 minutes) for tests run from Android Studio and the Google Developer Console, and 1500 seconds (25 minutes) for tests run from the gcloud command line.

App start-up timeout errors

If your app takes a long time to start, Robo can throw an error, and won't be able to crawl your app. This only happens in cases of extremely long start-up time, and can only be resolved by revising your app to make it start faster.

More control with Robo scripts

Sometimes you need more control over your tests. For example, you might want to test a common user journey or provide specific UI input like a username and password. Robo scripts can help.

With Robo scripts, you record yourself walking through a workflow in your app, then upload that recording to the Firebase console to run in Robo tests. When you run a Robo test with a script attached, Robo first steps through your pre-scripted actions then proceeds to explore the app as usual.

Get started with Robo scripting through the Firebase tool in Android Studio:

Open Android Studio.

In the main menu, select Tools > Firebase.

Select Test Lab and click Record Robo Script and use it to Guide Robo Test.

Follow the rest of the steps in the tool to record your Robo Script.

Upload that Robo Script to Test Lab and start your test.

Robo script errors

If a Robo script fails at any point, Test Lab abandons all further steps in the script, and resumes a regular Robo crawl. Most often, Robo scripts fail because Test Lab isn't able to find a necessary element on the screen. To avoid failures, make sure that your app navigation is predictable and that your screens aren't shown in some non-deterministic order.

Predefined text input

You can provide custom input text for other text fields used by your app. To provide text input for additional fields, do the following:

On the Select dimensions page, choose Additional options.

Under Additional fields (Optional), enter one or more resource names, and the strings to enter in the corresponding text fields.

The screenshot shows a configuration interface for Firebase Test Lab. It is divided into two main sections: 'Test account credentials (optional)' and 'Robo directives (optional)'. The first section has a link 'Learn more' and two input fields: 'edit_text_email' with the value 'venkat@gmail.com' and 'edit_text_password' with the value 'venkat'. The second section also has a 'Learn more' link and a list of four directives. Each directive consists of a dropdown menu set to 'Enter text', a resource name field, and a text input field. The directives are: 1. Resource 'EditTextEmail' with value 'ajay@ajay.com', 2. Resource 'EditTextPassword' with value 'ajayajay', 3. Resource 'edit_text_Email' with value 'venkat@gmail.com', and 4. Resource 'edit_text_Password' with value 'venkat'. Each row has a delete icon (an 'x' in a circle) to its right.

Predefined text input errors

Robo searches for EditText fields with an Android resource name that matches a supplied regular expression. If Robo can't find a matching field, it doesn't input your text, but otherwise continues its crawl as usual.

Analyze Firebase Test Lab Results

There are multiple ways to use Firebase Test Lab to run tests on your Android app, including the command line interface, Android Studio, the Test Lab UI in the Firebase console, and the Testing API. However you choose to start your tests, the results are stored in the Firebase project that you specify. You can explore the results using ToolResults API in addition to any of the tools above. This page describes how to review and analyze these test results.

KEY CONCEPTS

To see the results from all your previous test runs, select **Test Lab** in the left navigation panel of your project in the Firebase console. This page displays all the test runs from the apps that you have tested with your project using Test Lab.

To review test results, you first need to understand three concepts:

When you run a test or a set of test cases against devices and configurations you've selected, Test Lab runs the test against your app in a batch, then displays the results as a **test matrix**.

Devices x Test Executions = Test Matrix



Device

A physical or virtual device (Android only) you run a test on, such as a phone, tablet, or wearable device. Devices in a test matrix are identified by device model, OS version, screen orientation, and locale (also known as geography and language settings).



Test, test execution

A test (or a set of test cases) to be run on a device. You can run one test per device, or optionally shard the test and run its test cases on different devices.

Test matrix

Contains the statuses and test results for your test executions. If any test execution in a matrix fails, the whole matrix fails.

Test matrix	Test type	Started	Total devices	Status
matrix-xtzp1we6eqlja	Robot	1 day ago	5	---
matrix-27ur1ypw4d83	Robot	1 day ago	5	---
matrix-14ag3a4fyfag	Robot	1 day ago	5	---
matrix-31aw4ag3046	Robot	1 day ago	5	1 device crashed
matrix-8p3218ny56a3	Robot	1 day ago	5	---
matrix-374p2u3uk2w4	Robot	1 day ago	5	---
matrix-3k3d3e4ek4b7	Robot	1 day ago	5	---
matrix-1wag3a4fyfag	Robot	2 days ago	5	1 device crashed

The following sections explain how to navigate test results.

Interpret test history results

When you navigate to your test results by selecting **Test Lab**, you see the results of tests you have run so far.

Testing history is grouped by app. Only the most recent five test matrices are shown for each app; if more are available, you can click the **All Matrices** link at the bottom of the app test list to see the complete list for that app.

Interpret test matrix results

When starting a test through the Test Lab UI, you are redirected to a page where you can see your test matrix and click a specific test execution to view test results. Android Studio and the gcloud command provide a URL for the test matrix results page as well. In a typical test matrix, you might run a test across a dozen or so different devices. Each test execution can have a different outcome. The possible outcomes for any test execution in a test matrix include the following:

- ✓ • Passed : No failures were encountered.
- ! • Failed : At least one failure was encountered.
- ⚠ • Inconclusive : Test results were inconclusive, possibly due to a Test Lab error.
- ⊘ • Skipped : The selected dimension values for some test executions in the matrix were incompatible. This occurs when devices that you selected are incompatible with one or more of the Android API levels that you selected.

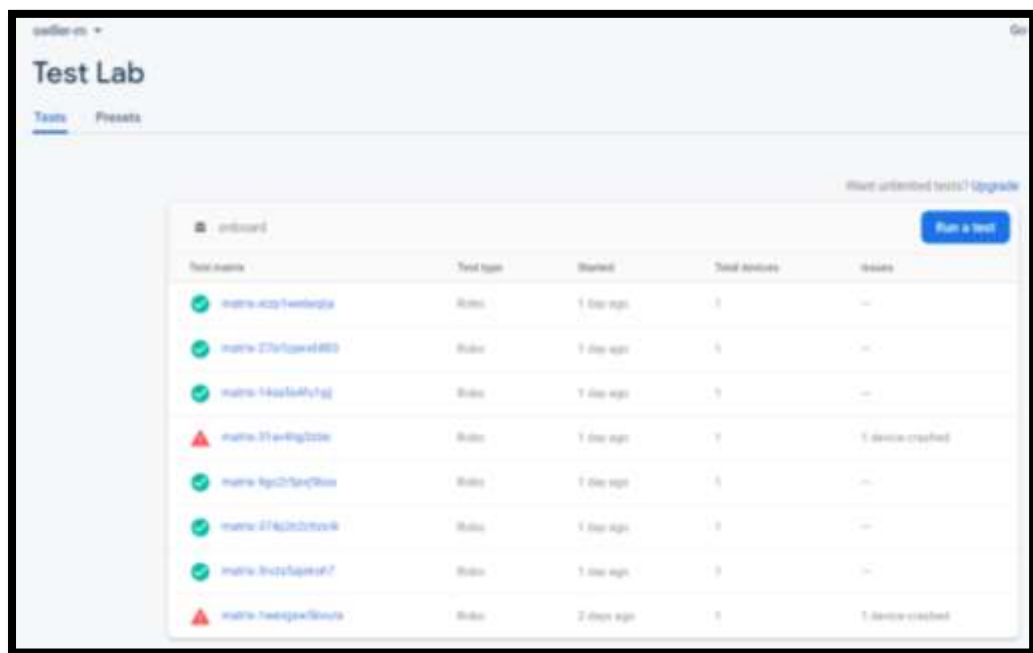
To review aggregated test results for all test matrices for a given app in your Firebase project, click the name of the app, as shown in the following example:

Example test matrix results page with only four test executions



This takes you to the test matrix list for your app, where you can click the name of any test matrix to see the test matrix results, and where you can click the name of the app (shown in the red box below) to view the test matrix list for other apps associated with your Firebase project.

Example test matrix list page



The screenshot shows the 'Test Lab' interface with a list of test matrices. The table has columns for Test matrix, Test type, Status, Total devices, and Issues. A 'Run a test' button is visible in the top right corner.

Test matrix	Test type	Status	Total devices	Issues
matrix-402f7e6b4a1a	Robot	1 day ago	5	---
matrix-270a7e6b4a1a	Robot	1 day ago	5	---
matrix-14a3b4b41a1a	Robot	1 day ago	5	---
matrix-270a7e6b4a1a	Robot	1 day ago	5	1 device crashed
matrix-402f7e6b4a1a	Robot	1 day ago	5	---
matrix-270a7e6b4a1a	Robot	1 day ago	5	---
matrix-14a3b4b41a1a	Robot	1 day ago	5	---
matrix-270a7e6b4a1a	Robot	2 days ago	5	1 device crashed

A test matrix can pass, fail, or be inconclusive. A test matrix is shown as failed or inconclusive if any test executions in that matrix fail or are inconclusive.

Interpret Robo test results

If you ran your tests with Robo, your results include videos and screenshots of Robo crawling your UI, in addition to the usual test metrics. Those video and screenshots include visual indications of the actions Robo took during the crawl, similar to the 'Show touches' feature in Android. You can use the indications to help you follow along with Robo's progress, and reproduce any bugs it might uncover.

Interpret results from a single test execution

From the test matrix results page, click on one of the test executions to see the result of that specific test execution.

Passed

10/31/21, 8:12 PM

5 min 17 sec

Portrait

English (United States)

Test results

Test issues

Robo

Logs

Screenshots

Videos

Performance

Accessibility

App start time

Time to initial display

239ms

Time to full display

Graphics state

Mouse Vsync

0%

High input latency

15%

Slow UI thread

0%

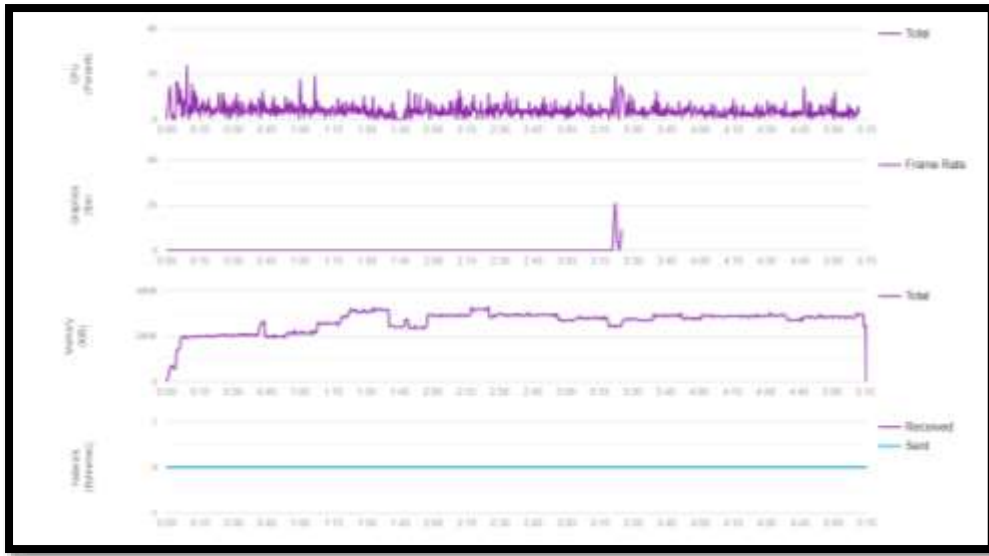
Slow draw commands

0%

Slow draw updates

0%

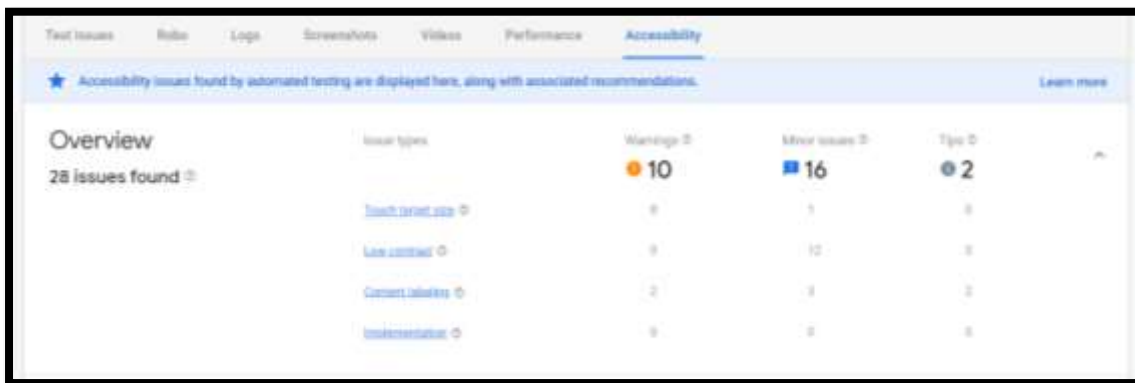
Distribution of UI render time



Example test execution results page

Interpret accessibility results

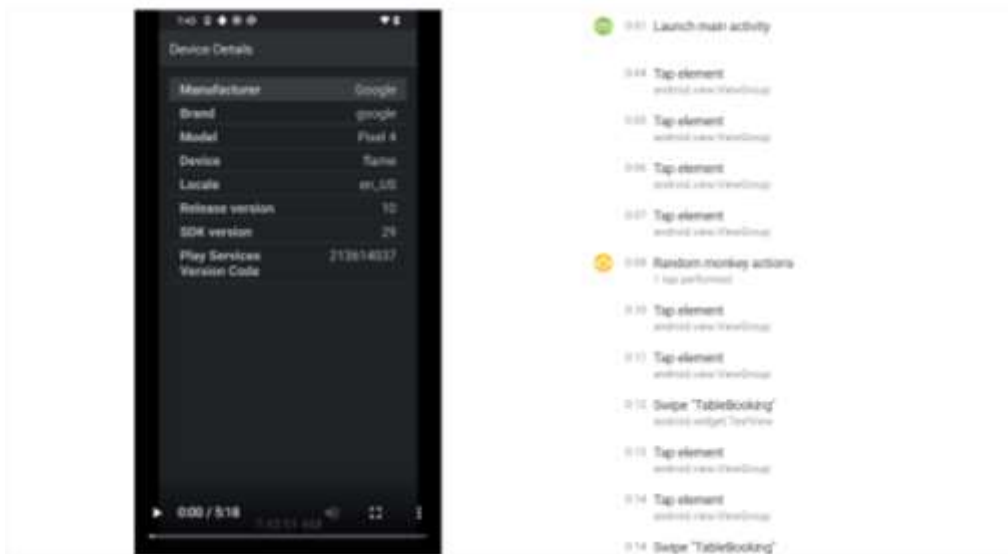
Robo tests use Android Accessibility Scanner to detect accessibility issues in your app (note that you can also run a scan locally on your device). For instructions on how to review and interpret the accessibility results of your Robo test, visit [Get started with Accessibility Scanner](#).



For general information on how to improve the accessibility of your app, visit the [Android Developer Accessibility documentation](#).

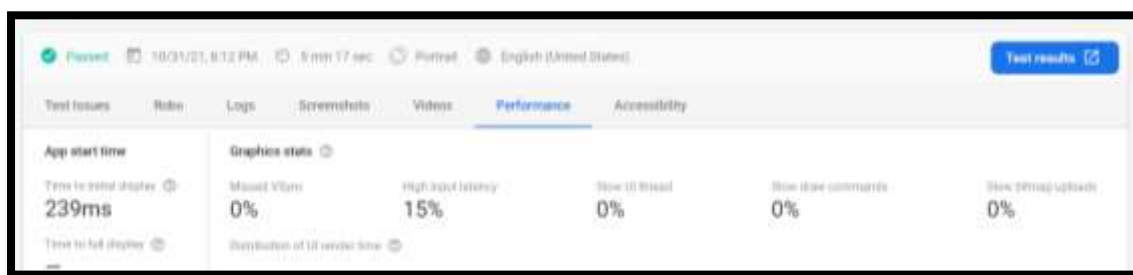
Performance metrics

Tests run on physical devices also return performance metrics:



The graphics performance report contains stats on several key graphics metrics:

- **Missed Vsync:** The number of missed Vsync events, divided by the number of frames that took longer than 16 ms to render.
- **High input latency:** The number of input events that took longer than 24 ms, divided by the number of frames that took longer than 16 ms to render.
- **Slow UI thread:** The number of times the UI thread took more than 8 ms to complete, divided by the number of frames that took longer than 16 ms to render.
- **Slow draw commands:** The number of times that sending draw commands to the GPU took more than 12 ms, divided by the number of frames that took longer than 16 ms to render.
- **Slow bitmap uploads:** The number of times that the bitmap took longer than 3.2 ms to upload to the GPU divided by the number of frames that took longer than 16 ms to render. Render time: The distribution of render times for each frame of the test run. Render times greater than 32 milliseconds cause a perceptible slowdown of your UI. Render times of 700+ indicate frozen frames. Render data is gathered from dumsys graphicsstats.



Bugs Fixed:

The following screenshots shows some of the bugs that have occurred in the initial stage of testing and we have also mentioned the way in which we have overcome those bugs.

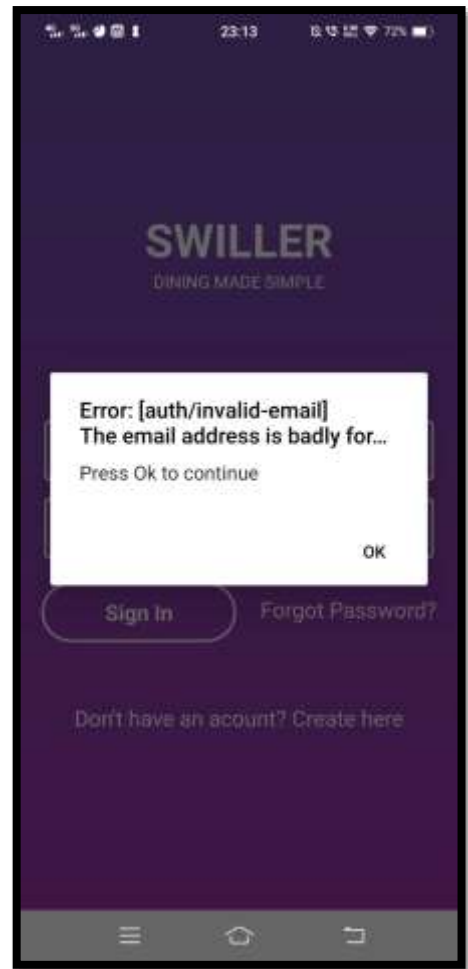


Bug: New user is created even when email of invalid format is provided.

Fix Method: Providing an alert whenever a user enters email address of invalid format.

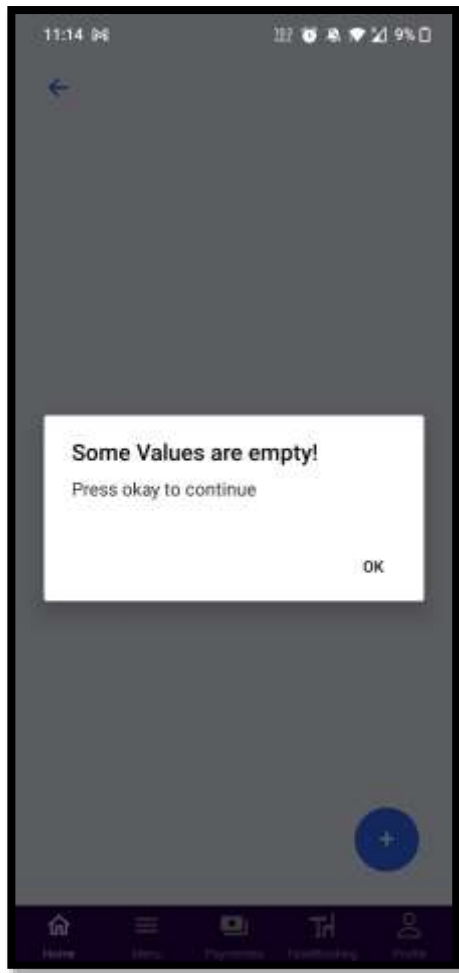
valid: abc@xyz.com

invalid: abc



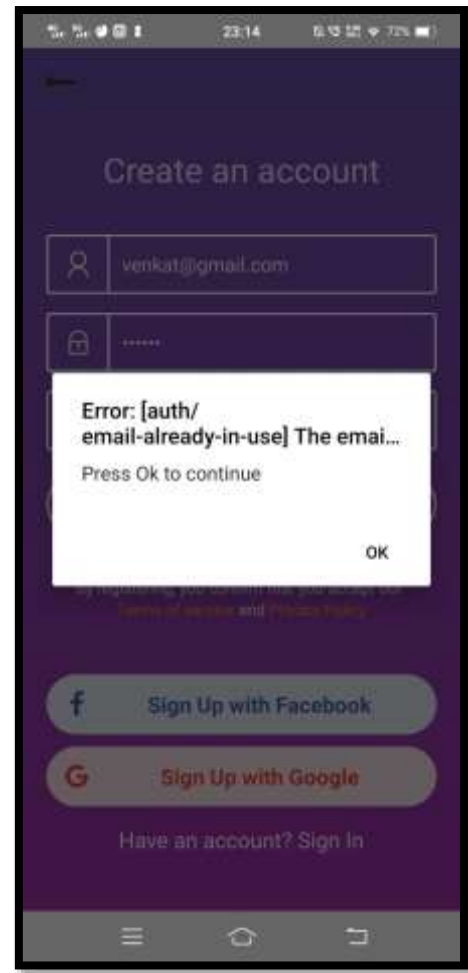
Bug: Nothing is shown when an email is given in wrong format during sign-in.

Fix Method: Providing an alert when email of wrong format is given that allows user to change and give input accordingly.



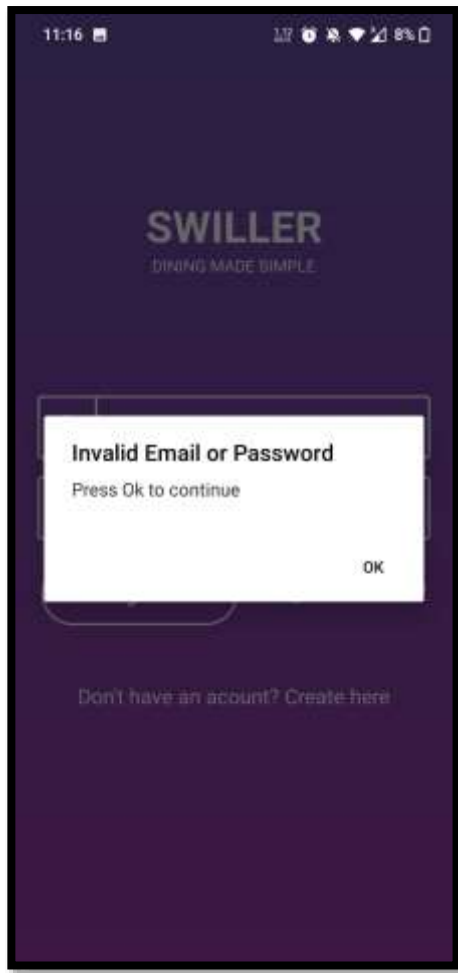
Bug: The staff and menu gets updated when provided with empty values.

Fix Method: Alert gets popped up when all the input boxes are not filled.



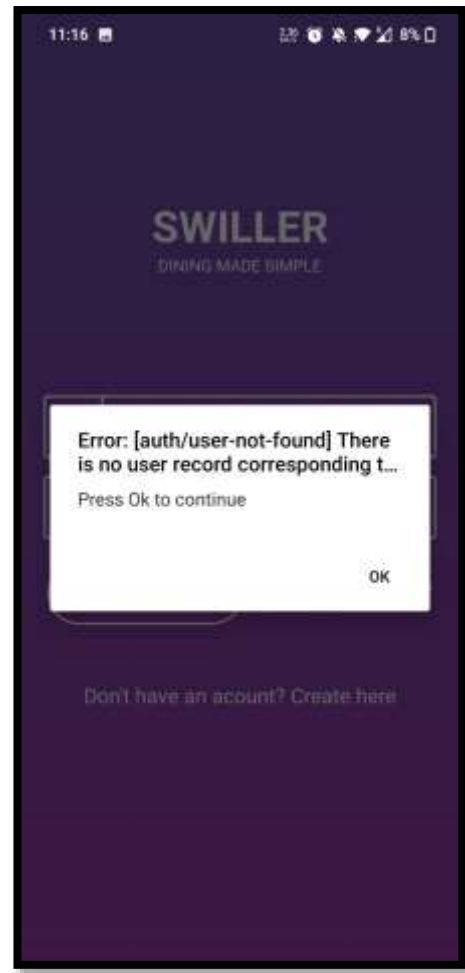
Bug: Users gets multiple accounts with same email address

Fix Method: Alert gets popped when a user creates an account with already existing email address thus allowing single account for each email address



Bug: Nothing is shown when user clicks on sign in without entering anything in the inputs boxes.

Fix Method: Alert gets popped up when nothing is provided in the input during sign in



Bug: Nothing is shown when user provides email that is not present in the database during sign in

Fix Method: Alert gets popped when the email provided is not found in the database.



Bug: Nothing is rendered when invalid email or password is provided as input.

Fix Method: Alert is popped when user provides invalid email or password during sign in.

Bug: Password gets accepted even it consists of single letter.

Fix Method: Alert gets popped when a weak password is chosen while creating account. A password containing atleast 6 characters must be given.

SECURITY UNDER CONSIDERATION

By authenticating your users and writing security rules, you can fully restrict read / write access to your Firebase data.

In a nutshell, Firebase security is enforced by server-side rules, that you author, and govern read or write access to given paths in your Firebase data tree.

Firebase security rules are JavaScript-like expressions: easy-to-write expressions that have access to the credentials for the connection, and the view of the Firebase data tree as it exists, along with pending changes on write.

In most cases, your client-side logic, templates, assets, etc. will be static and public. What you're really looking to secure is user and application data, and this is where Firebase Authentication (whether using custom Firebase authentication tokens or Firebase Simple Login) comes in. Firebase Authentication is essentially token generation - taking confirmed, identifiable user data and passing it securely to Firebase so that it cannot be spoofed. This confirmed credential data is then made available in your security rules.

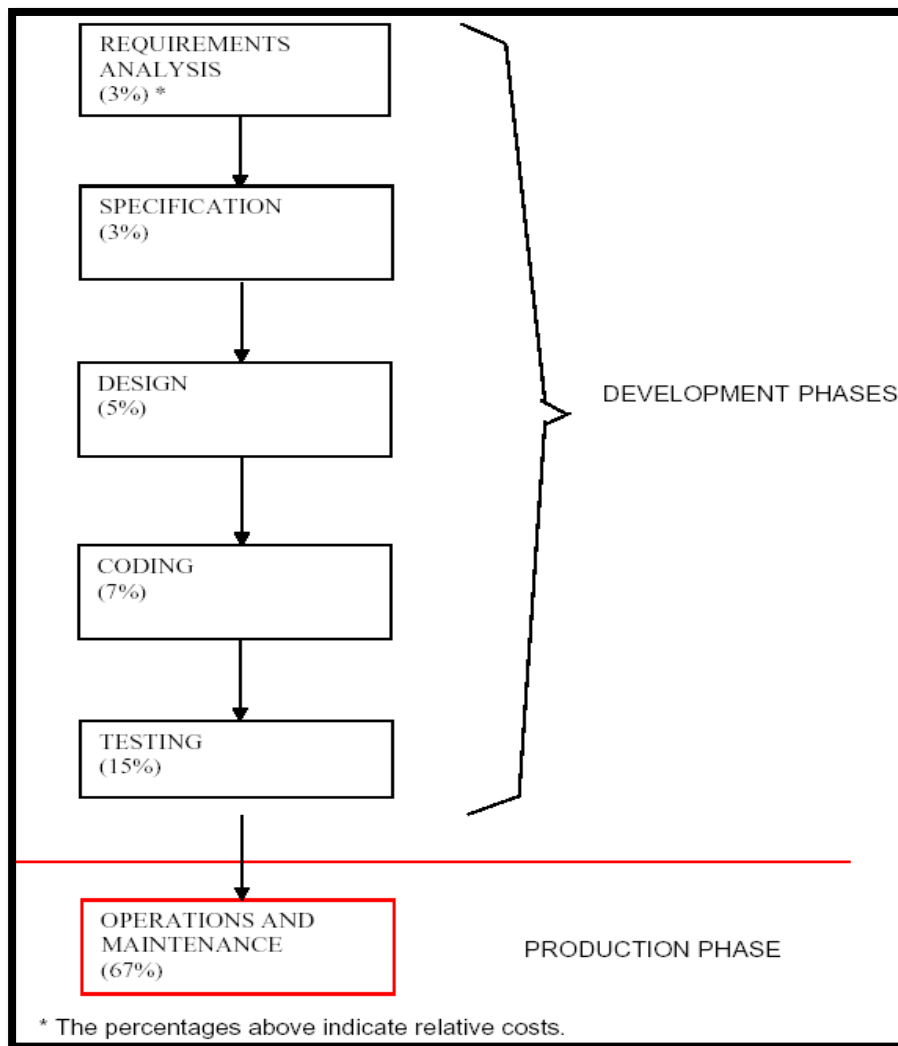
MAINTENANCE

INTRODUCTION

In software engineering, software maintenance is the process of enhancing and optimizing deployed software (software release), as well as remedying defects. Software maintenance is one of the phases in the software development process, and follows deployment of the software into the field. The software maintenance phase involves changes to the software in order to correct defects and deficiencies found during field usage as well as the addition of new functionality to improve the software's usability and applicability.

Software maintenance involves a number of specific techniques. One technique is static slicing, which is used to identify all the program code that can modify some variable. It is generally useful in refactoring program code.

The software maintenance phase is an explicit part of the waterfall model of the software development process which was developed during the structured programming movement of computer programming. The other major model, the spiral model developed during the object oriented movement of software engineering makes no explicit mention of a maintenance phase. Nevertheless, this activity is notable, considering the fact that two-thirds of a software system's lifetime cost involves maintenance.



In a formal software development environment, the developing organization or team will have some mechanisms to document and track defects and deficiencies. Software just like most other products, is typically released with a known set of defects and deficiencies. The software is released with the issues because the development organization decides the utility and value of the software at a particular level of quality outweighs the impact of the known defects and deficiencies.

The people involved in the software maintenance phase are expected to work on these known issues, address them, and prepare for a new release of the software, known as a maintenance release, which will address the documented issues.

Nature of Maintenance

Software maintenance sustains the software product throughout its operational life cycle. Modification requests are logged and tracked, the impact of proposed changes is determined, code and other software artifacts are modified, testing is conducted, and a new version of the software product is released. Also, training and daily support are provided to users. Pfleeger states that “maintenance has a broader scope, with more to track and control” than development.

Maintainers can learn from the developer’s knowledge of the software. Contact with the developers and early involvement by the maintainer helps reduce the maintenance effort. In some instances, the software engineer cannot be reached or has moved on to other tasks, which creates an additional challenge for the maintainers. Maintenance must take the products of the development, code, or documentation, for example, and support them immediately and evolve/maintain them progressively over the software life cycle.

Need for Maintenance

Maintenance is needed to ensure that the software continues to satisfy user requirements. Maintenance is applicable to software developed using any software life cycle model (for example, spiral). The system changes due to corrective and non-corrective software actions. Maintenance must be performed in order to:

- Correct faults
 - There are minor bugs to be fixed in the application including some error handlers and authentication issues.
 - These bugs need to be resolved before the users find them.
- Improve the design
 - The design of our application must be improved a bit more to catch user attention.
 - Using animations and graphic designs provides a good look to the application.
- Implement enhancements
 - Some new features can be added to the application such as home delivery, categories in payment, staff recruitment etc.,

- Interface with other systems
 - We have to make the app more responsive and test using many different models and screen ratios to make sure that the app doesn't crash in any of the used device models.
 - Since, we have developed application for android, we can also make it work in ios devices also.
- Migrate legacy software
 - We released the version 1 of the software which is till not outdated. So, this point doesn't come into view in near time.

Majority of Maintenance Costs

Maintenance consumes a major share of software life cycle financial resources. A common perception of software maintenance is that it merely fixes faults. However, studies and surveys over the years have indicated that the majority, over 80%, of the software maintenance effort is used for non-corrective actions. Jones describes the way in which software maintenance managers often group enhancements and corrections together in their management reports. This inclusion of enhancement requests with problem reports contributes to some of the misconceptions regarding the high cost of corrections. Understanding the categories of software maintenance helps to understand the structure of software maintenance costs. Also, understanding the factors that influence the maintainability of a system can help to contain costs. Some of the technical and non-technical factors affecting software maintenance costs, as follows:

- Application type
- Software novelty
- Software maintenance staff availability
- Software life span
- Hardware characteristics
- Quality of software design, construction, documentation and testing

CATEGORIES OF MAINTENANCE

I) Corrective Software Management:

Corrective software maintenance is what one would typically associate with the maintenance of any kind. Correct software maintenance addresses the errors and faults within software applications that could impact various parts of your software, including the design, logic, and code. These corrections usually come from bug reports that were created by users or customers but corrective software maintenance can help to spot them before your customers do, which can help your brand's reputation.

- Our application contains some minor bugs out of which majority of them were rectified and some are yet to be taken care of.
- Those minor bugs will not affect the users mostly. Once the app gets released, the user feedbacks will let us know about some issues and we try to rectify them.

2) Adaptive Software Maintenance:

Adaptive software maintenance becomes important when the environment of your software changes. This can be brought on by changes to the operating system, hardware, software dependencies, Cloud storage, or even changes within the operating system. Sometimes, adaptive software maintenance reflects organizational policies or rules as well. Updating services, making modifications to vendors, or changing payment processors can all necessitate adaptive software maintenance.

- Creating a real payment portal when the application is set to release for general users would help the users to pay online.
- Any further changes and implementations can be made based on the user requests when the app gets released.³

3) Perfective Software Maintenance:

Perfective software maintenance focuses on the evolution of requirements and features that existing in your system. As users interact with your applications, they may notice things that you did not or suggest new features that they would like as part of the software, which could become future projects or enhancements. Perfective software maintenance takes over some of the work, both adding features that can enhance user experience and removing features that are not effective and functional. This can include features that are not used or those that do not help you to meet your end goals.

- Our application consists of some asynchronous actions such as retrieving data from database especially images, uploading images both in staff list and menu list. This can be rectified using a skeleton UI that shows the default component while the original components are being retrieved.
- When the user flow increases, the staff and the orders list gets increases so it becomes difficult to sort them, so a search bar must be implemented to make the work easier.

4) Preventive Software Maintenance:

Preventative Software Maintenance helps to make changes and adaptations to your software so that it can work for a longer period of time. The focus of the type of maintenance is to prevent the deterioration of your software as it continues to adapt and change. These services can include optimizing code and updating documentation as needed.

Preventative software maintenance helps to reduce the risk associated with operating software for a long time, helping it to become more stable, understandable, and maintainable.

- We have made sure that most of the technologies used in the software are up to date so this issue will not come in near time.
- But, the firebase database that we have chosen is a free version that has validity only until 90 days. So, for longer usage of data base a premium cloud storage has to be created that handles the database for longer time.

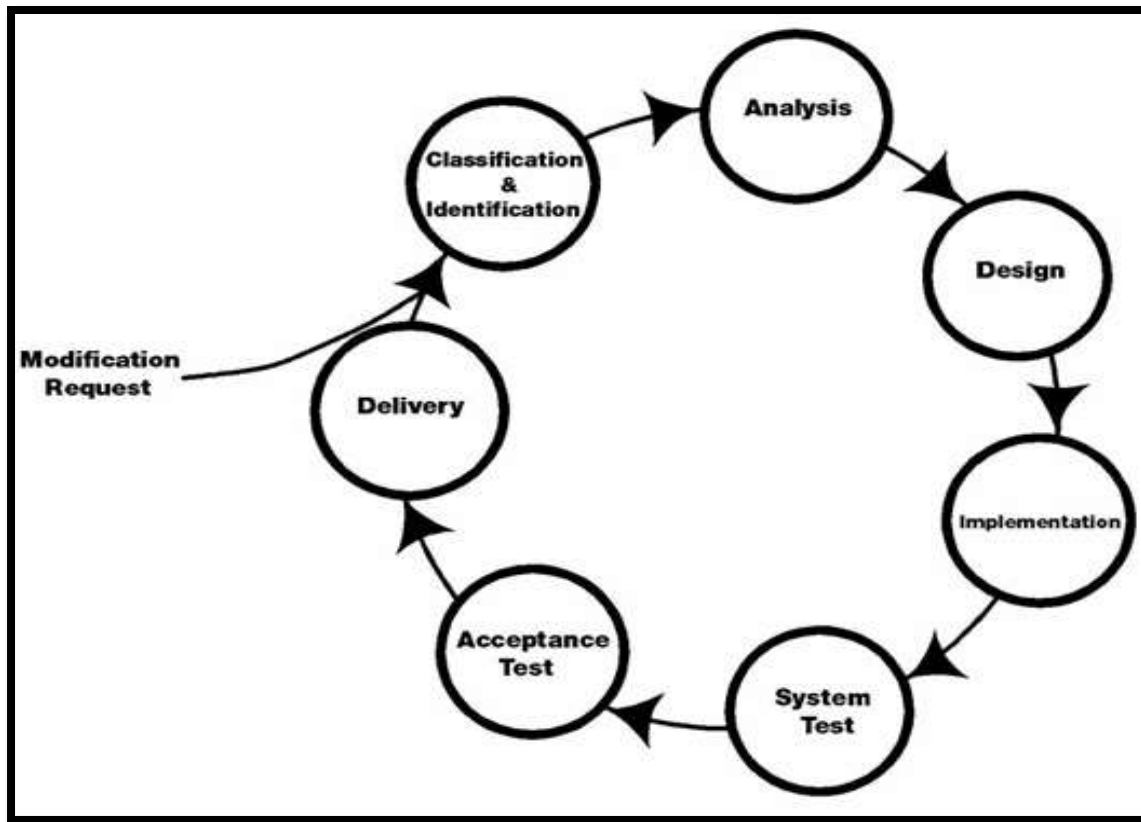
Key Issues in Software Maintenance

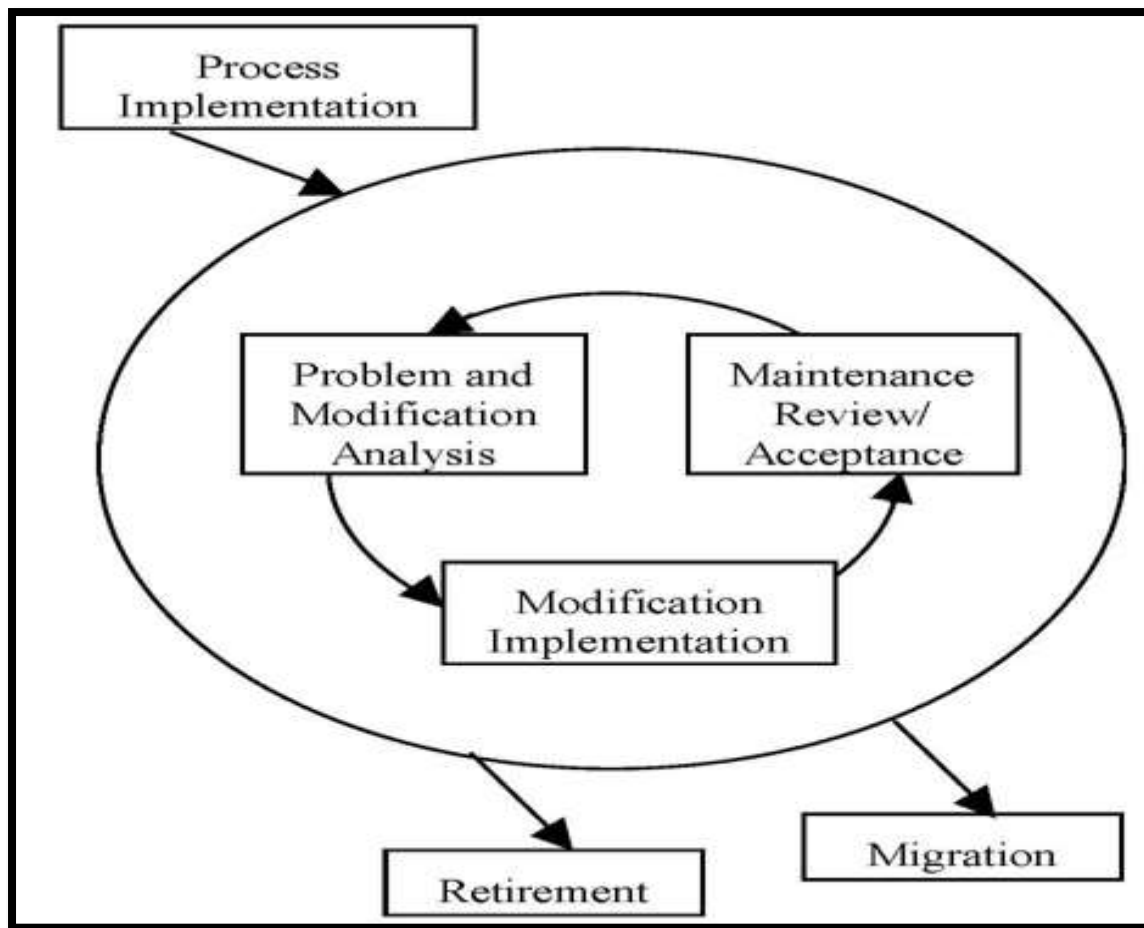
A number of key issues must be dealt with to ensure the effective maintenance of software. It is important to understand that software maintenance provides unique technical and management challenges for software engineers. Trying to find a fault in software containing 500K lines of code that the software engineer did not develop is a good example. Similarly, competing with software developers for resources is a constant battle. Planning for a future release, while coding the next release and sending out emergency patches for the current release, also creates a challenge. The following section presents some of the technical and management issues related to software maintenance. They have been grouped under the following topic headings:

- **Technical issues**
 - We have rectified majority of the issues that occurred during app testing.
 - We can have some more once the app is set to release. They will get rectified based on the user requests.
- **Management issues**
 - Asynchronous actions must be taken care because it takes a while for the data to retrieve from database. This can get rectified by implementing a loading screen when the data is being retrieved or updated.
- **Cost estimation and Measures**
 - We will face cost issues in creating premium cloud storage that helps our application to run longer with real-time updates.
 - Other costs include server maintenance costs and few software purchases.

Maintenance Process

The Maintenance Process subarea provides references and standards used to implement the software maintenance process. The Maintenance Activities topic differentiates maintenance from development and shows its relationship to other software engineering activities.





Each of the primary software maintenance activities is further broken down into tasks, as follows.

- Process Implementation
- Problem and Modification Analysis
- Modification Implementation
- Maintenance Review/Acceptance
- Migration
- Software Retirement

Maintenance planning activity

An important activity for software maintenance is planning, and maintainers must address the issues associated with a number of planning perspectives:

- Business planning (organizational level)
- Maintenance planning (transition level)
- Release/version planning (software level)
- Individual software change request planning (request level)

At the individual request level, planning is carried out during the impact analysis. The release/version planning activity requires that the maintainer:

- Collect the dates of availability of individual requests
- Agree with users on the content of subsequent releases/versions

- Identify potential conflicts and develop alternatives
- Assess the risk of a given release and develop a back-out plan in case problems should arise
- Inform all the stakeholders

Whereas software development projects can typically last from some months to a few of years, the maintenance phase usually lasts for many years. Making estimates of resources is a key element of maintenance planning. Those resources should be included in the developers' project planning budgets. Software maintenance planning should begin with the decision to develop a new system and should consider quality objectives (IEEE1061-98). A concept document should be developed, followed by a maintenance plan.

The concept document for maintenance should address:

- The scope of the software maintenance
- Adaptation of the software maintenance process
- Identification of the software maintenance organization
- An estimate of software maintenance costs

Finally, at the highest level, the maintenance organization will have to conduct business planning activities (budgetary, financial, and human resources) just like all the other divisions of the organization.