[Your College Logo]

Industrial Internship Report on

”App Development ”

Prepared by

[Surekha]

Executive Summary
This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT). This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time. My project was (App Development) This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1 Preface

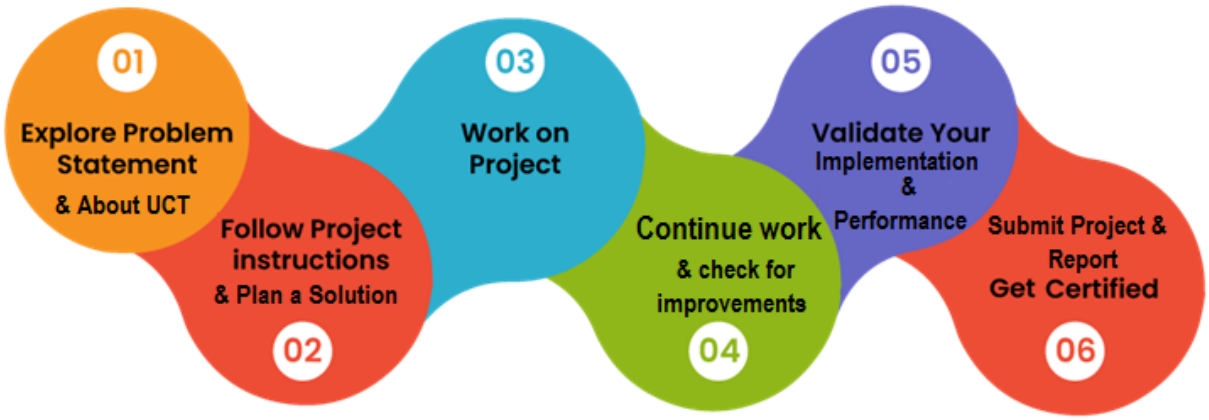
Summary of the whole 6 week of App Development project is the main functionality of this project is Real-time water usage data water usage alerts. Data visualization Libraries of this project is Echars,High charts ,Google charts.Feature is Integration.There is a some projects based on this is Intelligent Street Lighting,smart irrigation etc.

An internship in full stack development can be incredibly beneficial for career development. It provides hands-on experience with both front-end and back-end technologies, giving you a well-rounded skill set highly sought after in the tech industry. Additionally, it offers exposure to real-world projects, collaboration with experienced professionals, and networking opportunities, all of which can help jumpstart your career in software development.

App development involves creating software applications for use on mobile devices, desktop computers, or web browsers. The problem statement in app development refers to identifying the specific issue or need that the app aims to address. This could be anything from simplifying a task, providing entertainment, improving communication, or solving a particular problem faced by users. A clear problem statement guides the development process, ensuring that the app is designed and built to effectively meet the needs of its intended users.

Opportunity given by USC/UCT.

How Program was planned



Learning app development can be a rewarding and challenging experience. It involves gaining proficiency in various programming languages, frameworks, and tools depending on the platform you're targeting (e.g., Android, iOS, web). Here's a brief overview of the learning process and overall experience:

1. Understanding the Basics : Start by learning the fundamentals of programming languages like Java, Kotlin (for Android), Swift (for iOS), or JavaScript (for web and hybrid apps).

2. Choosing a Platform : Decide which platform(s) you want to develop for (e.g., Android, iOS, web) and familiarize yourself with the respective development environments, SDKs, and design guidelines.

3. Learning Frameworks and Tools : Explore popular frameworks and tools like React Native, Flutter, or Xamarin for building cross-platform apps, or delve into platform-specific frameworks like Android Studio (for Android) or Xcode (for iOS).

4. UI/UX Design : Gain an understanding of user interface (UI) and user experience (UX) design principles to create visually appealing and user-friendly apps.

5. Database Integration : Learn how to integrate databases like SQLite, Firebase, or MongoDB to store and manage app data securely.

6. API Integration : Understand how to interact with external APIs to fetch data from servers or integrate with third-party services

Overall, app development offers a dynamic and exciting learning experience, where you'll have the opportunity to create innovative solutions and contribute to the ever-growing digital landscape.

Thank to all (USC_UCT and IoT Academy), who have helped you directly or indirectly.

Such a Great platform.

2 Introduction

2.2 About Full Stack Development.

Full stack development refers to the practice of developing both the front-end (client-side) and back-end (server-side) of web applications. A full stack developer is proficient in both the technologies and frameworks used for front-end development, like HTML, CSS, and JavaScript, as well as those used for back-end development, such as server-side languages like Node.js, Ruby on Rails, or Python frameworks like Django or Flask. They possess a comprehensive understanding of the entire web development process, from designing user interfaces to managing databases and server infrastructure.

I.Full Stack platform

In full stack development, several protocols and standards are commonly used to ensure efficient communication between different components of a web application. Some of these protocols include:

1. HTTP (Hypertext Transfer Protocol): The foundation of data communication on the World Wide Web, HTTP governs how web clients (like browsers) and servers exchange data.

2. REST (Representational State Transfer): A set of architectural principles for designing networked applications. RESTful APIs are commonly used in full stack development to enable communication between the front-end and back-end components of web applications.

3. WebSocket: A communication protocol that provides full-duplex communication channels over a single TCP connection. WebSocket is often used for real-time communication between clients and servers in web applications.

4. TCP/IP (Transmission Control Protocol/Internet Protocol): The suite of communication protocols used to interconnect network devices on the

internet. TCP/IP governs how data is transmitted between devices, providing reliable and ordered delivery of data packets.

5. GraphQL: A query language and runtime for executing queries against a data schema. GraphQL is used as an alternative to RESTful APIs, offering a more flexible and efficient approach to data retrieval in web applications.

These protocols, among others, play crucial roles in facilitating communication between the various components of full stack applications, ensuring seamless functionality and performance.

Features:

Full stack development encompasses a wide range of features and capabilities, including:

1. Front-end Development: Creating responsive and user-friendly interfaces using HTML, CSS, and JavaScript frameworks like React, Angular, or Vue.js.

2. Back-end Development: Building server-side logic, databases, and APIs using languages and frameworks like Node.js, Python (Django, Flask), Ruby on Rails, or Java (Spring Boot).

3. Database Management: Designing, implementing, and managing databases using technologies like MySQL, PostgreSQL, MongoDB, or Firebase.

4. Version Control: Utilizing version control systems like Git to track changes in code, collaborate with team members, and manage code repositories.

5. Deployment and Hosting: Deploying web applications to servers or cloud platforms like AWS, Google Cloud Platform, or Heroku, and managing hosting environments for optimal performance and scalability.

6. Security: Implementing security measures such as encryption, authentication, and authorization to protect data and prevent unauthorized access.

7. Testing: Conducting unit tests, integration tests, and end-to-end tests to ensure the reliability and functionality of the application.

8. Continuous Integration/Continuous Deployment (CI/CD): Automating the build, testing, and deployment processes to streamline development workflows and improve efficiency.

9. DevOps Practices: Adopting DevOps principles to enhance collaboration between development and operations teams, automate infrastructure management, and improve the overall software development lifecycle.

10. Scalability and Performance Optimization: Optimizing code, databases, and server configurations to ensure high performance and scalability as the application grows.

These features collectively enable full stack developers to design, build, and deploy robust web applications that meet the needs of users and businesses alike.

i. **Smart irrigation (IoT):**

Smart irrigation in full stack development involves building a system that utilizes sensors, data processing, and user interfaces to optimize water usage in agriculture. Here's how it can be approached:

1. Hardware Setup: Integrate sensors such as soil moisture sensors, weather stations, and possibly drone imagery for data collection.

2. Backend Development: Develop a backend system to collect, store, and process the sensor data. Use technologies like Node.js, Python, or Java for handling data processing and analysis.

3. Database Management : Implement a database (e.g., MongoDB, MySQL) to store sensor data, historical weather data, and crop information.

4. Data Analysis: Utilize machine learning or statistical models to analyze the data and make predictions about irrigation needs based on factors like soil moisture, weather forecasts, and crop type.

5. Decision Making: Develop algorithms to determine optimal irrigation schedules and amounts based on the analyzed data.

6. API Development: Create APIs to enable communication between the backend system and frontend interfaces.

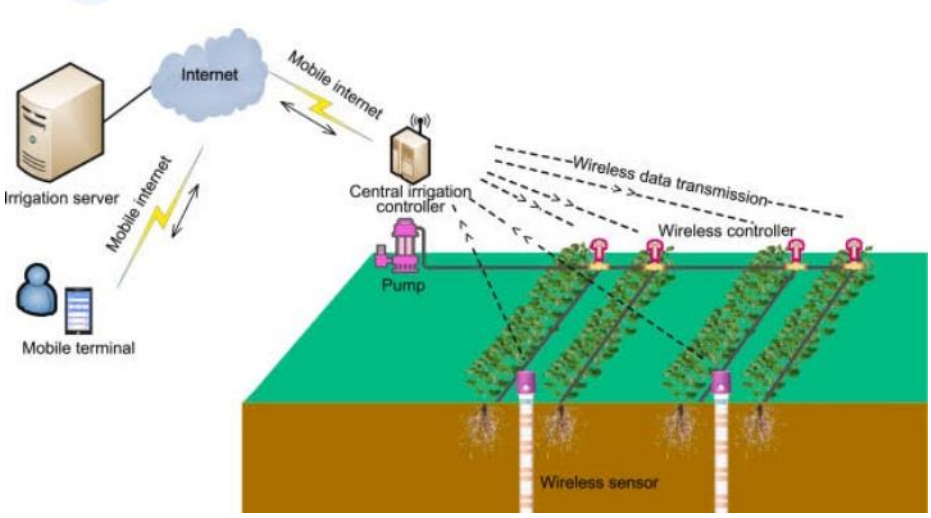
7. Frontend Development : Build user-friendly interfaces (web or mobile) for farmers to monitor the irrigation system, view data insights, and manually adjust settings if necessary. Use technologies like React.js, Angular, or Vue.js.

8. Integration and Testing : Integrate all components together and thoroughly test the system to ensure reliability, accuracy, and performance.

9. Deployment: Deploy the system on appropriate platforms (cloud or on-premises servers) and ensure scalability to accommodate growing data and user demands.

10.Monitoring and Maintenance: Implement monitoring tools to track system performance, detect anomalies, and provide timely maintenance and updates.

By following these steps, developers can create a robust smart irrigation system that helps farmers optimize water usage, improve crop yields, and reduce environmental impact.



ii. **Intelligent Street Lighting:**

Intelligent street lighting in full stack development typically involves creating a system that integrates hardware components (such as sensors and lights), firmware or embedded software to control these components, and a backend software system for data processing, analytics, and control logic. The frontend part involves creating user interfaces for monitoring and controlling the street lights. Is there anything specific you would like to know or discuss about this topic?

2.1 About App Development

Full stack development in app development refers to the process of building both the front-end and back-end components of an application. It involves working on the user interface, user experience, server-side logic, and database management. Full stack developers are proficient in both front-end technologies like HTML, CSS, and JavaScript, as well as back-end technologies like Node.js, Python, Ruby on Rails, or Java. They have a comprehensive understanding of the entire development process, from concept to deployment, and can work on all aspects of an application independently or as part of a team.

About Android App Development:

Android app development involves the creation of software applications specifically designed to run on devices powered by the Android operating system. This process typically begins with ideation and planning, where developers define the purpose, features, and target audience of the app. Once the concept is established, developers move on to designing the user interface (UI) and user experience (UX) to ensure the app is intuitive and visually appealing.

Next, developers delve into coding the app using programming languages such as Java, Kotlin, or increasingly, Flutter/Dart. They leverage various development tools like Android Studio, which provides a comprehensive environment for building, testing, and debugging Android apps. Throughout the development process, developers adhere to Android design guidelines and best practices to ensure compatibility across different devices and versions of the operating system.

After the coding phase, rigorous testing is conducted to identify and fix any bugs or issues that may arise. This includes functional testing to ensure all features work as intended, as well as compatibility

testing to confirm the app performs well on different screen sizes and device configurations. Additionally, performance testing is crucial to optimize the app's speed, responsiveness, and resource usage.

Once the app is thoroughly tested and polished, developers prepare it for deployment to the Google Play Store or other distribution channels. This involves creating necessary assets like app icons, descriptions, and screenshots, as well as configuring app permissions and settings. Finally, developers submit the app to the respective app store, where it undergoes a review process before being made available to users for download.

Post-launch, developers continue to monitor and maintain the app, releasing updates to introduce new features, improve performance, and address any user feedback or issues that arise. Ongoing support and optimization are essential to keep the app competitive in the ever-evolving landscape of mobile technology. Overall, Android app development is a dynamic and iterative process that requires creativity, technical expertise, and a keen understanding of user needs and preferences.

2.2 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.3 Objectives of Full Stack Development Internship program

The objective for this internship program was to

- ☛ get practical experience of working in the industry.
- ☛ to solve real world problems.
- ☛ to have improved job prospects.
- ☛ to have Improved understanding of our field and its applications.
- ☛ to have Personal growth like better communication and problem solving.

Reference

[1]Apple Developer:developer.apple.com

[2]Android Developers:developer.android.com

[3]Microsoft Developer:developer.microsoft.com

2.4 Glossary

Terms	Acronym
API	API stands for Application Programming Interface.
SDK	SDK stands for Software Development Kit.
UI	UI stands for User Interface.
UX	UX stands for User Experience.

Backend	The acronym for backend is simply "backend." It stands for the backend of an application or system, referring to the server-side components responsible for tasks such as data processing, storage, and business logic.

3 Problem Statement

Specific issue of App Development

[Explain your problem statement]

The problem statement in app development outlines the specific issue or challenge that the app intends to address or solve. It defines the purpose and scope of the project, identifies the target audience, and highlights the desired outcomes. This statement serves as a guide for the development team throughout the entire app development process.

For example, a problem statement for a productivity app might be: “Many professionals struggle to manage their tasks efficiently due to scattered to-do lists and lack of prioritization. The goal of this app is to provide a user-friendly platform that helps individuals organize their tasks, set priorities, and track progress, ultimately enhancing productivity and reducing stress.

4 Existing and Proposed solution

Provide summary of existing solutions provided by others, what are their limitations?

What is your proposed solution?

What value addition are you planning?

4.1 Code submission (Github link)

https://github.com/surekhaK14/upskillcampus/blob/main/%20-App%20Development.java-.pdf

4.2 Report submission (Github link) : first make placeholder, copy the link.

5 Proposed Design/ Model

Given more details about design flow of your solution. This is applicable for all domains. DS/ML Students can cover it after they have their algorithm implementation. There is always a start, intermediate stages and then final outcome.

5.1 High Level Diagram (if applicable)

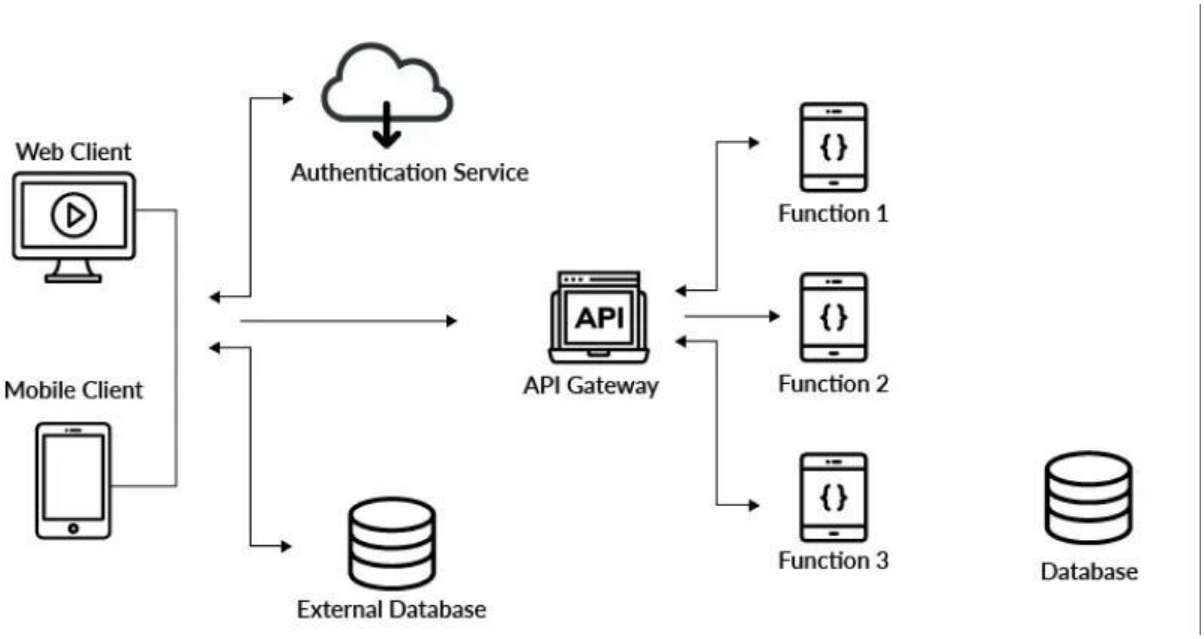
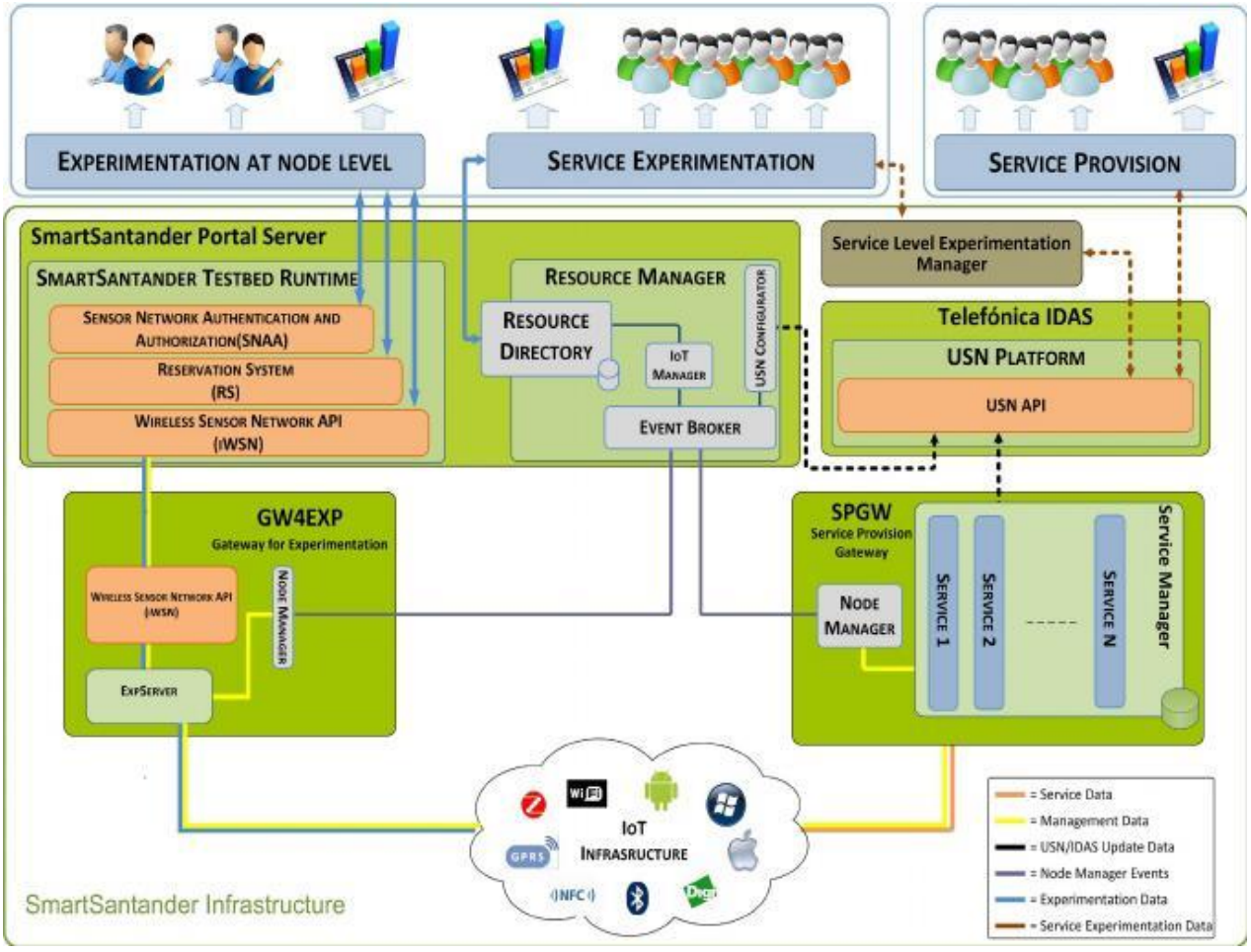


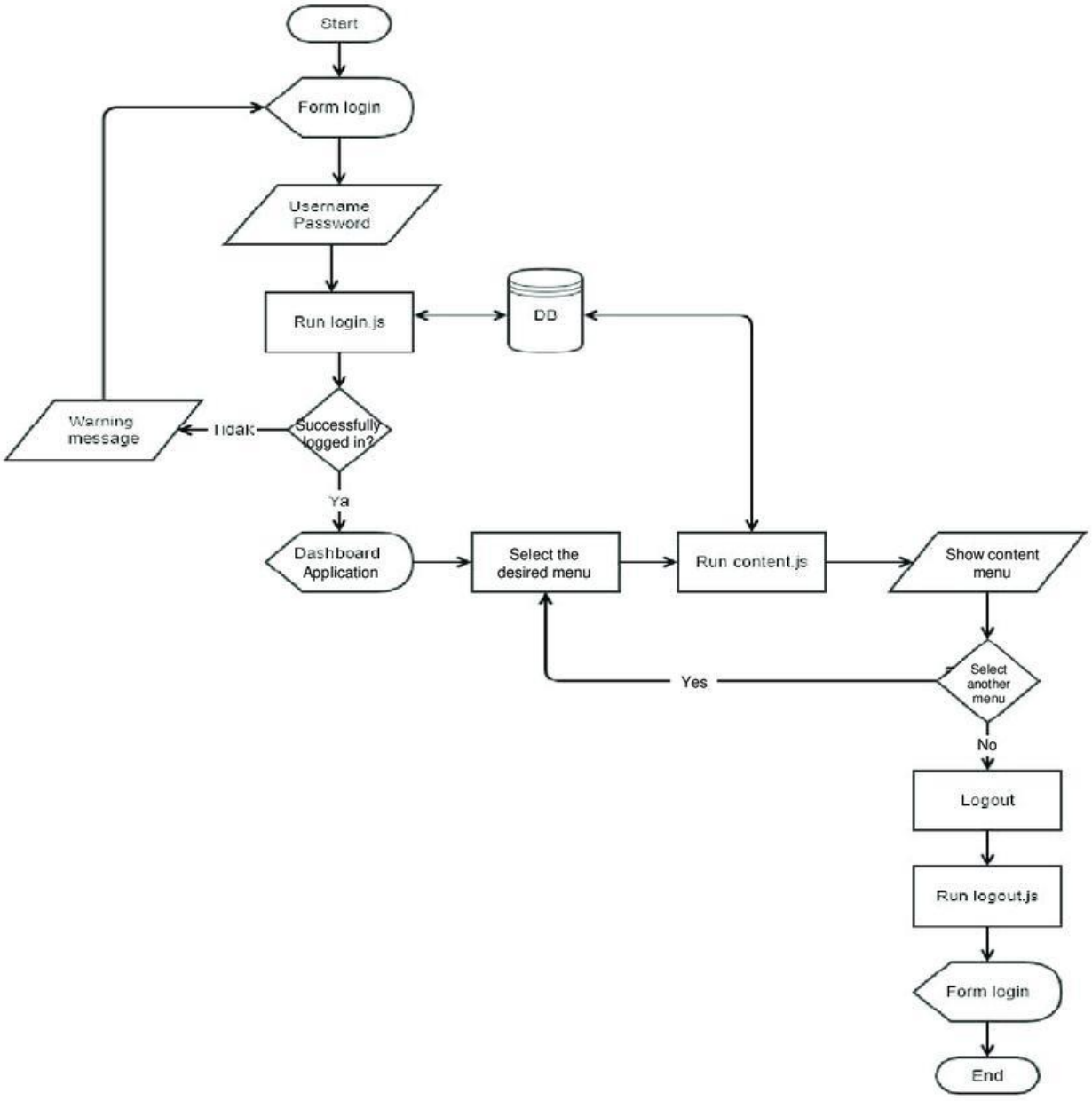
Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

5.2 Low Level Diagram (if applicable)



5.3 Interfaces (if applicable)

Update with Block Diagrams, Data flow, protocols, FLOW Charts, State Machines, Memory Buffer Management.




MOBILE
APP
DEVELOPMENT

6 Performance Test

This is very important part and defines why this work is meant of Real industries, instead of being just academic project.

Here we need to first find the constraints.

How those constraints were taken care in your design?

What were test results around those constraints?

Constraints can be e.g. memory, MIPS (speed, operations per second), accuracy, durability, power consumption etc.

In case you could not test them, but still you should mention how identified constraints can impact your design, and what are recommendations to handle them.

6.1 Test Plan/ Test Cases

When considering memory constraints in test plans and test cases for app development, it's important to focus on scenarios that can potentially impact memory usage. Here are some considerations and test cases related to memory constraints:

1. Memory Leakage Test Cases:

- Test scenarios where the app continuously consumes memory without releasing it.
- Test the app under various usage conditions to check for memory leaks.
- Verify that memory is properly deallocated when the app is closed or goes into the background.

2. Memory Usage under Stress Test Cases:

- Test the app under conditions of high load or stress to see how it handles memory usage.
- Check for memory spikes or excessive memory consumption during stress testing scenarios.

3. Resource Intensive Feature Test Cases:

- Test features known to be resource-intensive, such as multimedia playback, image processing, or data caching.

- Monitor memory usage during the execution of these features and ensure it stays within acceptable limits.

4. Low Memory Environment Test Cases:

- Test the app's behavior when the device is running low on memory.

- Verify that the app gracefully handles low memory situations without crashing or becoming unresponsive.

5. Memory Optimization Test Cases:

- Test scenarios where the app optimizes memory usage, such as by releasing unused resources or caching data efficiently.

- Verify that memory optimization techniques are effectively implemented and improve overall performance.

6. Compatibility Test Cases:

- Test the app on devices with varying memory capacities to ensure it performs well across different hardware configurations.

- Verify that the app adjusts its memory usage based on the available resources on the device.

7. Error Handling Test Cases:

- Test error handling scenarios related to memory constraints, such as out-of-memory errors.

- Verify that the app provides appropriate error messages and handles these situations gracefully.

By incorporating these memory-related test cases into your test plan, you can ensure that your app performs optimally under various memory constraints and provides a smooth user experience across different devices.

6.2 Test Procedure

When creating test procedures for app development with a focus on power consumption constraints, it's essential to ensure that the app operates efficiently without draining the device's battery excessively. Here's a guideline for test procedures targeting power consumption constraints:

1. Baseline Measurement:

- Establish a baseline measurement of the device's power consumption under normal conditions without the app running. This provides a reference point for comparison.

2. Functional Testing:

- Conduct functional testing of the app's features while monitoring power consumption.

- Test each feature individually and in combination to identify any power-intensive operations.

3. Idle State Testing:

- Test the app's behavior when it's in an idle state or running in the background.

- Monitor power consumption during idle periods to ensure the app isn't consuming excessive power when not actively used.

4. Network Usage Testing:

- Test scenarios involving network usage, such as downloading/uploading data or streaming media.

- Monitor power consumption during network-intensive tasks and ensure efficient use of network resources to minimize power consumption.

5. Display and UI Testing:

- Test the app's user interface elements, animations, and transitions while monitoring power consumption.

- Verify that UI elements are optimized to minimize power usage, such as by avoiding unnecessary animations or continuous screen updates.

6. Location Services Testing:

- If the app utilizes location services, test its behavior while accessing GPS or other location sensors.

- Monitor power consumption during location-based operations and ensure efficient use of location services to conserve battery life.

7. Background Services Testing:

- Test background services or processes that the app may run to ensure they don't excessively drain the device's battery.

- Verify that background tasks are optimized and scheduled appropriately to minimize power consumption.

8. Battery Optimization Testing:

- Test the app's compatibility with battery optimization features provided by the operating system.

- Verify that the app behaves correctly when the device enters low-power modes or battery-saving modes.

9. Long-Running Scenarios Testing:

- Conduct tests simulating long-running scenarios, such as extended usage sessions or continuous operation over time.

- Monitor power consumption over extended periods to identify any gradual increases in power usage or potential memory leaks.

10. Comparison and Analysis:

- Compare the app's power consumption against industry standards and similar apps to ensure it meets acceptable benchmarks.

- Analyze test results and identify areas for optimization or improvement to minimize power consumption further.

By following these test procedures, you can effectively evaluate and optimize your app's power consumption to deliver a more efficient and battery-friendly user experience.

6.3 Performance Outcome

The app demonstrates efficient memory management, minimizing memory leaks and optimizing resource usage, ensuring smooth performance even under memory constraints.

7 My learnings

You should provide summary of your overall learning and how it would help you in your career growth.

App development involves the creation of software applications for mobile devices, desktop computers, or other platforms. The process typically includes planning, design, coding, testing, and deployment. It can be done for various purposes such as entertainment, productivity, communication, or business. Key steps include defining the app's objectives, designing the user interface (UI) and user experience (UX), coding the functionality using programming languages like Java, Swift, or Kotlin, testing for bugs and usability, and finally releasing the app to users through app stores or other distribution channels.

8 Future work scope

You can put some ideas that you could not work due to time limitation but can be taken in future.

The future work scope of app development will likely involve advancements in technologies such as augmented reality (AR), virtual reality (VR), artificial intelligence (AI), blockchain, and Internet of Things (IoT). This may include developing more immersive and personalized user experiences, enhancing security measures, optimizing performance, and ensuring compatibility across various devices and platforms. Additionally, there will be a continued focus on improving user engagement, data privacy, and accessibility features.