People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdès

Institute of Electrical and Electronic Engineering
Department of Electronics

Project Report Presented in Partial Fulfilment of

the Requirements of the Degree of

'Master'

In Computer Engineering

Title:

Development of a Mobile Application for the Operational Interruption Centre Information

Presented By:

- **OULD BABA ALI Hanane**

Supervisor:

Dr. A. BENZEKRI

Co-Supervisor:

- **Mr. A. AMARA (ELIT)**

Registration Number:...../2024

Abstract

This report describes the design and development of a mobile application for the Operational Interruption Center Information, OICI system for short. This system is a vital part of Sonelgaz Group's infrastructure, Algeria's national electricity and gas company.It is Aimed at streamlining the management of operational interruptions and enhancing real-time communication, the OICI system centralizes data and improves decision-making. The project has successfully delivered a robust solution that integrates key technologies such as Python Django Rest Framework for backend development, React Native for mobile frontend,and PostgreSQL for database management, enhancing efficiency and responsiveness. Also reported the design process, challenges faced, solutions implemented, and future enhancement. This highlights Sonelgaz's commitment to innovation and excellence in service delivery.

Acknowledgement

In the name of Allah, the Most Gracious and the Most Merciful

Alhamdulillah, all praises to Allah for the strengths and His blessing in completing this thesis.

I am deeply grateful to my supervisor,Dr. A.BENZEKRI, whose precious knowledge and guidance have been invaluable throughout this endeavor.

I extend my heartfelt appreciation to my co-supervisor, Mr. AMARA, who provided assistance during my training and proposed this project.

To all my family members—Mom, Dad, Sisters Racha and Merieme, and Brother Merouane ,thank you for always being there for me, through my best and worst moments. I hope I have made you proud.

Special thanks to my childhood friends, Sawsen and Maya, whose unwavering support has been a constant source of comfort and joy for over 14 years. Words cannot express the depth of gratitude I feel for your enduring friendship.

To Imen, Omnia, Lamis, and Ichrak, my companions in the journey of university life, thank you for the joy and camaraderie. Without you, campus life wouldn't have held nearly as much value.

And to all my friends, who have shared both laughter and tears with me, thank you for being there through the ups and downs of life.

Dedication

To those who believed in me, this work is dedicated. Your unwavering support and encouragement have fueled my journey, inspiring me to push the boundaries of what is possible.

List of Figures

List of Tables

List of Abbreviations

Abbreviation	**Meaning**
SQL	Structured Query Language
OICI	Operational Interruption Center Information
API	Application Programming Interface
JS	JavaScript
CSS	Cascading Style Sheets
DRF	Django Rest Framework
REST	Representation state transfer
DOM	Document Object Model
UI	User Interface
UX	User experience
EAS	Expo Application Services
FBV	Function-Based Views
CBV	Class-Based Views
ORM	Object-Relational Mapping

Contents

Chapter 1

Introduction

1.1. Problematic

The realm of application development is ever-evolving, with innovative solutions continually emerging to address complex business needs. This report delves into the development of a mobile application designed for the Operational Interruption Center Information ,OICI, system, a pivotal component within the Sonelgaz Group's infrastructure. Sonelgaz, the national company for electricity and gas in Algeria, stands at the forefront of energy production and distribution, catering to millions of households across the nation.

The project's inception is rooted in the necessity to streamline the management of operational interruptions and enhance real-time communication within the Sonelgaz Group. By harnessing the power of mobile technology, the OICI system aims to centralize data, facilitate instant access to information, and improve decision-making processes at various hierarchical levels.

This comprehensive report outlines the theoretical background that underpins the application's development, including an overview of key technologies and methodologies that guided the project. It provides an in-depth analysis of the Sonelgaz Group's structure, the role of ELIT Sonelgaz in information systems, and the significance of SICIO in monitoring electric incidents. Furthermore, it explores the intricacies of back-end and front-end development, database management,Application Programming Interface ,API, design, and the importance of user interface and user experience design.

As we navigate through the report, we will uncover the systematic approach taken to design and implement the OICI system. From the architectural blueprint to the challenges encountered and the solutions adopted, each aspect of the project is meticulously documented. The report also casts light on future enhancements

that promise to elevate the application's functionality and user experience.

In essence, this project is not merely a testament to technological advancement but also a reflection of the Sonelgaz Group's commitment to innovation and excellence in serving its customers. The ensuing pages will provide a detailed account of the journey undertaken to bring this ambitious project to fruition.

1.2. Sonelgaz Group

Sonelgaz, (Société nationale de l'électricité et du gaz), national company of electricity and gas in english, is the leading group in the production and distribution of electricity and gas in Algeria. It was created in 1962, through Ordinance No. 69-59 of July 28, 1969, which dissolved the public establishment Electricité et Gaz d'Algérie (EGA). The group occupies a privileged position in the country's economy as responsible for supplying more than seven million households with electricity and three million with natural gas, representing a geographical coverage of nearly 99% for the electrification rate and 55% for gas penetration.

1.2.1. Organization of the Group

The Sonelgaz Group has implemented, for several years now, a new organization by opting for an industrial approach in the energy field. Currently, the Sonelgaz group is made up of the Sonelgaz holding company, 36 subsidiaries and 8 partner companies. These different entities are governed by a set of management bodies and are presented in fig 1.1

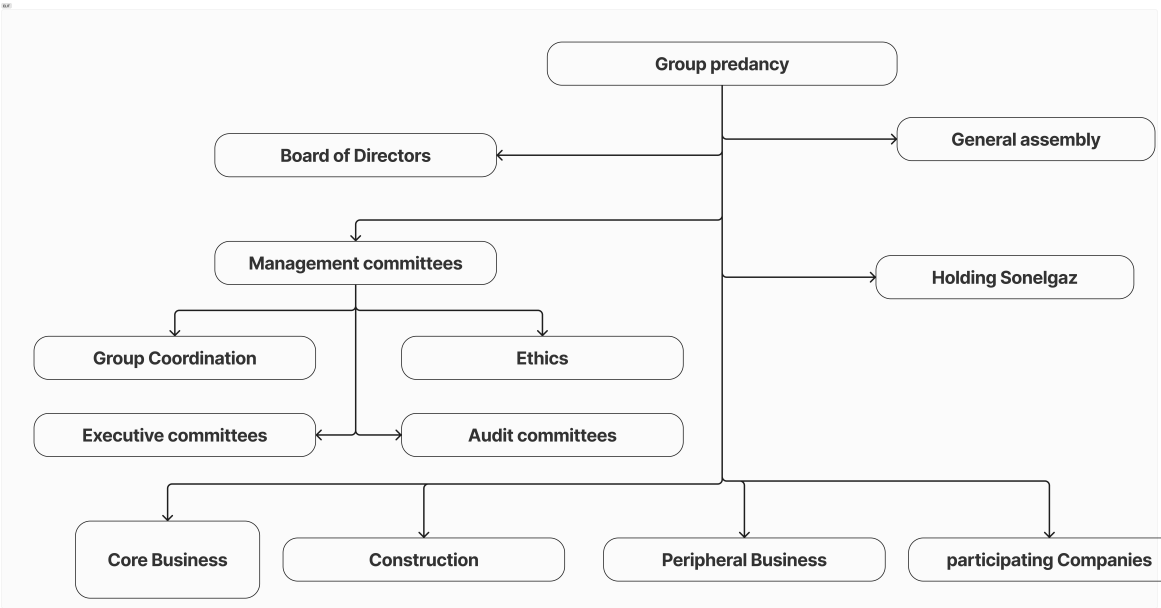


Fig. 1.1. Structure of Sonelgaz Group

- **Management Bodies:** The strategic decisions, as well as the monitoring of the various activities of the Sonelgaz group, are made by its different management bodies:
  - **Board of Directors:** A collegial management body. The Board of Directors (CA) controls the exercise of activities within the Sonelgaz Group and ensures the development of the objectives of its subsidiaries while enforcing the laws and regulations in force.
  - **General Assembly:** This is the absolute body where strategic decisions concerning the future of the group are made. It is composed of representatives of the state, the sole owner of the share capital.
  - **Management Committees:** These are committees that serve to control the management of the various companies in the group.
- **Group Companies:**
  - **Parent Company (Sonelgaz Holding):** As part of the new directions, the holding company ensures the steering of the group, through the exercise of policy and strategy missions.
  - **Core Business Subsidiaries:** There are twelve of them. These subsidiaries ensure: production, transport, and distribution of electricity, as well as transport and distribution of gas by pipeline.
  - **Construction Subsidiaries:** These are subsidiaries specialized in the field of energy infrastructure realization (engineering, industrial assembly, network realization...).
  - **Peripheral Business Subsidiaries:** They are in charge of ancillary activities, such as maintenance of energy equipment, distribution of electrical and gas equipment, transport, exceptional handling, research, development, training, etc.
  - **Participating Companies:** Sonelgaz also holds participations in companies, whose business is related to the field of electricity and gas.

1.2.2. Client Company SONELGAZ-DISTRIBUTION

The client company: SONELGAZ DISTRIBUTION spa, previously named SADEG, was established on 2017, with a share capital of 64 billion Dinars. It is responsible for the distribution of electricity and gas across the national territory and must ensure the continuity of its service by operating and maintaining its distribution networks. It includes 52 distribution directorates, 353 commercial agencies with 8,810,312 electricity customers and 4,921,959 gas customers, not less than 190

Electricity Districts and 181 Gas Districts. figure 1.2 illustrates in detail the organization of SD. A distribution directorate is a decentralized entity at the level of each wilaya, responsible for ensuring the distribution of energy, the operation of the energy infrastructure, and the creation of new networks to meet customer demands. A distribution region encompasses several distribution directorates and is in charge of supervising them.

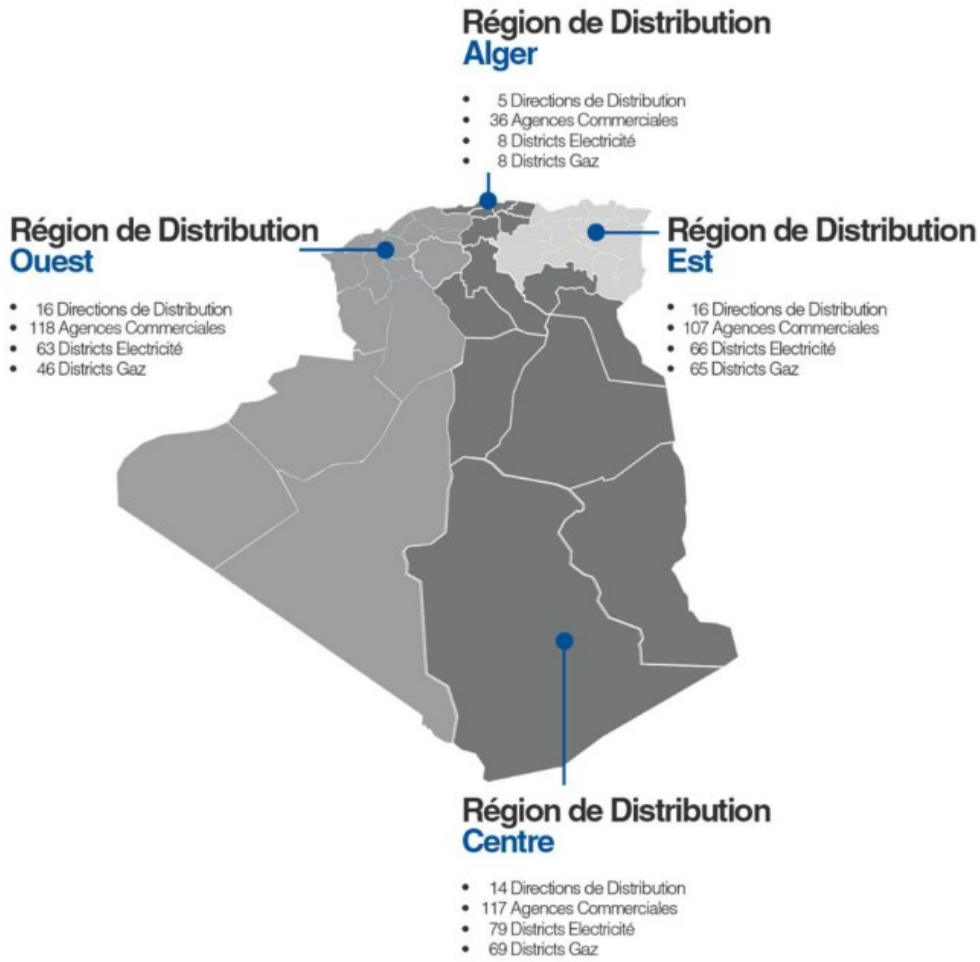


Fig. 1.2. Regions of Destrebutions

1.2.3. ELIT Sonelgaz

On January 1, 2009, the Information Systems activity, hitherto entrusted to the General Directorate of Information Systems at Sonelgaz, was erected into a joint-stock company, called "EL Djazair Information Technology", abbreviated "ELIT SPA". This subsidiary was created to respond to:

- The Sonelgaz Group's strategy of developing its own project management means in the field of Information Systems, and of having a technological skills center at the service of its companies.

- The Sonelgaz Group's desire to entrust the ownership of Information Systems to a specialized entity and to focus the capacities of its companies on their respective core businesses.

And like any other subsidiary of the Sonelgaz group, ELIT has its own mono-structure relatively independent of the other subsidiaries.
fig 1.3 show determine the structure of ELIT

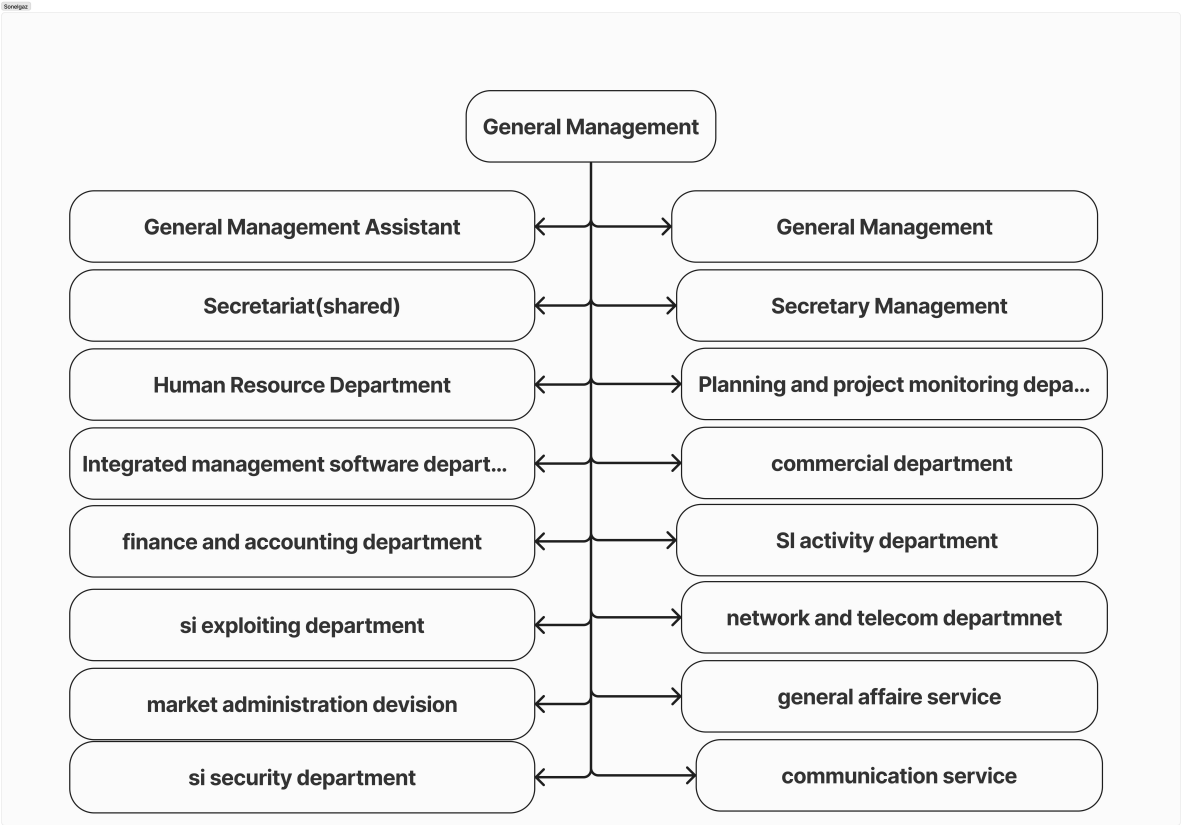


Fig. 1.3. Structure of Elit

1.2.4. Structure and organization of the "DSID" reception

Our end-of-study internship was carried out at the level of the IT direction Distribution Activities / Network Management abbreviated as DSID(direction SI activités distribution/gestion) . More precisely, at the level of the department IT distribution development. The DSID has three departments as shown in the organization chart 1.4:

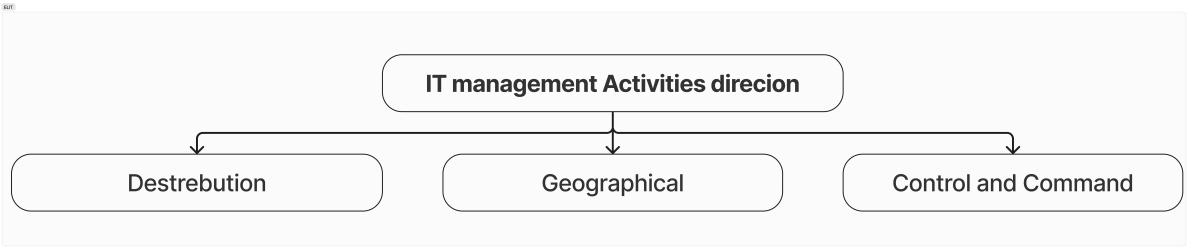


Fig. 1.4. Structure of DSID

The IT Department for Distribution Activities / Network Management is responsible for the production, deployment, and maintenance of information systems, distribution trades, and activities that involve the use of a network or are based on network infrastructures, which may include embedded and/or mobile computer systems. Among other things, it involves activities requiring the implementation of intelligent systems (Smart Grids) related to the transportation and distribution of electricity and gas.

1.2.5. Operational Interruption Information Systems

The SICIO (Système d'Information CIO) is a specialized module designed for the Centre d'Information Opérationnelle within Sonelgaz, aimed at automating the process of report generation and distribution. It addresses the logistical challenges faced in the daily dissemination of operational reports related to the electrical and gas systems. The SICIO module enhances the efficiency and quality of reports by automating their creation, validation, and distribution, including the provision of SMS alerts for significant events. This automation is crucial for timely and accurate information delivery to the executive management of DG/SADEG and other Sonelgaz group leaders, ensuring continuous availability and improved decision-making based on reliable data.

1.3. Report Organization:

Beside the Introduction in Chapter 1, Chapter 2 provides Theoretical Background of the project , while the System Design is well described in chapter 3, and we finished with chapter 4 as a conclusion Chapter.

Chapter 2

Theoretical Background

In this chapter, we present the theoretical foundation that underpins the development of our mobile application for the Operational Interruption Center Information. This includes an overview of the key technologies, and methodologies that guided our project.

2.1. What is Application Development

Application development, or app development, is the process of planning, designing, creating, testing, and deploying a software application to perform various business operations. It can be done by massive organizations with large teams working on projects or by a single freelance developer. Application development defines the process of how the application is made and generally follows a standard methodology.

2.2. Back-end Development

Back-end development refers to the work on the unseen aspects of a website, primarily focusing on server-side software. The role of back-end developers is to ensure the proper functioning of the website. They concentrate on elements such as databases, server logic, APIs, architecture, which are the link between front-end and back-end systems, and servers. They employ code that facilitates the interaction between browsers and databases, enabling data storage, interpretation, and deletion.

In a team setting, back-end developers work in conjunction with front-end developers, product managers, principal architects, and website testers to construct the framework of a website or mobile application. It's essential for back-end developers to be proficient in various tools and frameworks, including programming lan-

guages like Python, Java, and Ruby. Their primary responsibility is to ensure that the back-end responds swiftly and efficiently to requests from the front-end[1].

2.3. Database

A **database** is an organized collection of data, also referred to as structured data. This data can be accessed or stored in a computer system and managed through a **Database Management System, DBMS,**, a software used for data management. A database refers to related data in a structured form.

2.3.1. Types of Databases

There are various types of databases:

- **Relational Database:** A relational database consists of a set of tables with data fitting into a predefined category.
- **Distributed Database:** A distributed database is one where parts of the database are stored in multiple physical locations and processing is dispersed or replicated among different points in a network.
- **Cloud Database:** A cloud database typically runs on a cloud computing platform. Database services provide access to the database and make the underlying software stack transparent to the user.

These interactions exemplify a traditional database where data is of one type, i.e., textual. Technological advancements have led to new applications of database systems. New media technology has enabled the storage of images and video clips, leading to the creation of multimedia databases.

- **Object-Oriented Databases:** Similar to object-oriented programming, data in an object-oriented database is represented as objects.
- **Data Warehouses:** A data warehouse is a type of database created specifically for quick query and analysis. It serves as a central repository for data.
- **NoSQL Databases:** Unlike relational databases, which specify how all data input must be formatted, NoSQL, or nonrelational databases, allow the storage and manipulation of unstructured and semi-structured data.
- **Graph Databases:** Data is stored in a graph database using entities and their relationships.
- **OLTP Database:** An OLTP database is a fast, analytical database designed to handle numerous transactions from multiple users simultaneously.

- **Open source databases:** An open-source database system can have either a SQL or NoSQL database as its source code.
- **Cloud databases:** A cloud database is a collection of organized or unorganized data that is housed on a private, public, or hybrid cloud computing platform. Cloud database models come in two flavors: traditional and database as a service ,DBaaS,. With DBaaS, a service provider handles maintenance and administrative duties.

PostgreSQL

PostgreSQL is an open-source object-relational database management system ,ORDBMS for short, that inherits its foundation from the pioneering **POSTGRES** project developed at the University of California, Berkeley's Computer Science Department. Notably, POSTGRES introduced many concepts that weren't adopted by commercial database systems until much later.

As a descendant of the original Berkeley code, PostgreSQL adheres to a significant portion of the SQL standard while offering advanced features typically found in enterprise-grade solutions. These features include:

- **Complex Query Capabilities:** Facilitates intricate data retrieval through powerful querying mechanisms.
- **Relational Integrity Enforcement:** Enforces referential constraints through foreign keys, ensuring data consistency across tables.
- **Event-Driven Procedures (Triggers):** Enables automated execution of predefined actions upon specific database events (e.g., inserts, updates, deletes).
- **Updatable Views:** Allows for modifications to materialized views, simplifying data manipulation.
- **Transactional Integrity:** Guarantees the atomicity, consistency, isolation, and durability ,ACID, properties for database transactions.
- **Multiversion Concurrency Control (MVCC):** Provides a mechanism for optimistic concurrency control, minimizing locking conflicts and improving concurrency.

Furthermore, PostgreSQL boasts a high degree of user extensibility. Users can leverage its open-source nature to expand its capabilities by adding:

- **Custom Data Types:** Enables the creation of user-defined data types tailored to specific application needs.
- **User-Defined Functions, UDFs, :** Allows for the implementation of custom functions to extend PostgreSQL's functionality.

- **Custom Operators:** Facilitates the creation of custom operators to manipulate user-defined data types.
- **Aggregate Functions:** Enables the development of custom aggregate functions for efficient data summarization.
- **Index Methods:** Allows for the creation of user-defined indexing strategies to optimize query performance.
- **Procedural Languages:** Supports the integration of procedural languages like PL/pgSQL, enabling the development of complex database logic.

Finally, the liberal licensing terms associated with PostgreSQL permit unrestricted use, modification, and distribution for any purpose, including private, commercial, and academic endeavors[2].

2.4. Python Django Rest Framework

Python's Django REST Framework, DRF, is a powerful toolkit for building Web APIs. It is designed to work with Django, a high-level Python Web framework that encourages rapid development and clean, pragmatic design. DRF comes with a modular and customizable architecture, allowing developers to use its components independently or as a comprehensive framework.

At the core of DRF are **models**, which are essentially Python classes that define the structure of the database. Each model corresponds to a single database table, and each attribute of the model represents a database field. Models are used to create, read, update, and delete records in the database, following the Object-Relational Mapping, ORM, pattern.

Serializers in DRF play a crucial role in converting complex data types, like Django QuerySets and model instances, into Python data types that can then be easily rendered into JSON, XML, or other content types. They also provide deserialization, allowing parsed data to be converted back into complex types after validating the incoming data. The serializers work very similarly to Django's Form and ModelForm classes.

Views in DRF are responsible for handling the logic of processing a request and returning a response. There are two main patterns for writing views: Function-Based Views, FBV, and Class-Based Views ,CBV,. FBVs are simple functions that take a request and return a response, while CBVs are classes that allow developers to organize their code by grouping related behaviors and attributes into classes.

URLs in DRF are defined in the `urls.py` file, where URL patterns are associated

with views. This file acts as a mapping between URL patterns to the Python functions or classes that should be called when a certain URL is accessed.

Signals in Django and DRF are a form of event-driven programming. They allow certain senders to notify a set of receivers when certain actions have taken place. For example, a signal can be sent when a model is saved, allowing for additional logic to be executed in response to the save event.

Together, these components of DRF provide a robust set of tools for developers to build APIs that can handle complex data structures, user authentication, permissions, and more. With DRF, developers can focus on writing their application logic instead of reinventing the wheel for common tasks associated with Web API development.

2.5. What is an API

Application Programming Interfaces ,APIs for short, are the predominant method for linking users, applications, and services in today's IT landscape. Most contemporary applications are constructed using APIs, which are software interfaces that facilitate communication between applications or services, enabling interaction through requests and responses. Nevertheless, an increase in APIs also expands the potential for attacks. With the proliferation of APIs across microservices architectures, it becomes imperative to implement supplementary infrastructure to guarantee both scalability and security [3].

2.6. HTTP overview:

HTTP, or Hypertext Transfer Protocol, is a protocol used for retrieving resources like HTML documents. It forms the basis of any data transfer on the Web and operates on a client-server model. This means that the recipient, typically a Web browser, initiates requests. A complete document is assembled from various sub-documents that are fetched, such as text, layout descriptions, images, videos, scripts, and more.

In this model, clients and servers exchange distinct messages, rather than a continuous data stream. The messages that the client, usually a Web browser, sends are known as requests. The server sends messages in response to these requests, which are aptly called responses[4].

2.6.1. http methods:

HTTP methods and request types specify what the API endpoint should do with the resources. It defines the action. The API developer takes decisions and manipulates resources based on the HTTP methods specified in the HTTP request[5]. It can be summerized in the table 2.1

HTTP Method	Action
GET	Returns the requested resource. If not found, returns a 404 Not Found status code.
POST	Creates a record. The POST request always comes with an HTTP request body containing JSON or Form URL encoded data, which is also called a payload. If the data is valid, the API endpoint will create a new resource based on these data. Although you can create multiple resources with a single POST call, it is not considered a best practice to do so.
PUT	Instructs the API to replace a resource. Like a POST request, the PUT request also comes with data. A PUT request usually supplies all data for a particular resource so that the API developer can fully replace that resource with the provided data. A PUT request deals with a single resource.
PATCH	Tells the API to update a part of the resource. Note the difference between a PUT and a PATCH call. A PUT call replaces the complete resource, while the PATCH call only updates some parts. A PATCH request also deals with a single record.
DELETE	Instructs the API to delete a resource.

TABLE 2.1. HTTP methods and actions

2.6.2. response status codes:

- **100-199**:This range is used to convey information such as processing delays. For example, if an API needs time to process a request, it can return a 102 – Processing status until the result is ready. This informs the client to check back later for the final result[5].
- **200-299**:These codes indicate successful operations:
  200 - OK: For successful GET, PUT, PATCH, or DELETE operations.
  201 - Created: For successful POST operations where a new resource has been created[5].
- **300-399**:These codes indicate redirection. For example, if you change an API endpoint from /api/items to /api/menu-items, you can use a 301 - Moved Permanently status code to redirect clients to the new endpoint[5].
- **400-499**:4xx status codes are used in the following scenarios:
  404 - Not Found: When the client requests a non-existing resource.
  400 - Bad Request: When the client sends an invalid payload with insufficient data.
  401 - Unauthorized: When the client is not authenticated.

403 - Forbidden: When the client tries to perform an action it is not authorized for[5].

- **500- 599**:These status codes indicate server-side errors and are automatically generated if something goes wrong in the code and the API developer hasn't handled such errors. these errors should be avoided when creating the back-end of an application[5].

2.7. Mobile Application Development

Mobile application development involves creating software applications that run on mobile devices.

2.7.1. Operating Systems

The two dominant mobile operating systems, OS, are:

- **Android**: Developed by Google, Android is an open-source platform widely adopted due to its flexibility and extensive range of devices.
- **iOS**: Developed by Apple, iOS is known for its robust security features and seamless integration with other Apple products.

Each platform has its own development environment and tools, requiring developers to be familiar with both if they aim to target users across these operating systems.

2.8. Java Script

JavaScript, commonly abbreviated as JS, is renowned as the quintessential programming language for the web. Its utility transcends beyond merely facilitating dynamic client-side scripting on webpages; it has evolved to encompass server-side development through runtimes such as Node.js, and even extends to mobile application creation via frameworks like React. The contemporary landscape of software development would be markedly distinct in the absence of JavaScript's versatile contributions.

2.8.1. React

:React is a library in JavaScript used for constructing user interfaces.It simplifies the process of creating interactive UIs. It helps creating self-contained components that control their own state, and then combine them to build complex

UIs. Because the logic of the component is written in JavaScript rather than templates,developer can pass rich data through your app and keep the state away from the DOM.

Learn Once, Write Anywhere: React doesn't make assumptions about the rest of your technology stack, allowing you to develop new features in React without having to rewrite existing code. React can also render on the server using Node and can be used to build mobile apps with React Native[6].

React Hooks are functions that let you "hook into" React state and lifecycle features from function components. Here's a brief summary of the key Hooks in React:

- State Hooks: Allow components to "remember" information like user input or selected image.
- Effect Hooks: Enable components to interact with external systems, such as network requests or DOM updates.
- Context Hooks: is a potent feature that facilitates efficient data sharing within component trees, without the necessity to explicitly pass props down through each level. It provides a mechanism to bypass the cumbersome process of "prop drilling," where data is tediously passed from parent to child components, often through multiple layers that do not require the data themselves. By creating a context, developers can set a default value that any component can access, as long as it falls within the provider's scope. This is particularly useful for themes, user information, or in this case, heading levels in a UI structure. The context is established by wrapping the desired section of the component tree with a `Context.Provider` and passing the data through the `value` prop. Subsequently, any component that needs access to this data can utilize the `useContext` hook to subscribe to the context and react to any changes in the provided value. This mechanism is akin to CSS property inheritance, where styles are applied throughout a nested structure unless specifically overridden. In summary, React's context is an elegant solution for managing state and other data across a wide range of components with minimal prop passing.
- Ref Hooks: Hold information outside the component rendering flow.

These Hooks can be combined or used to create custom Hooks for more complex logic. Remember, Hooks are for function components and should not be used inside loops, conditions, or nested functions.

2.8.2. React Native

React Native applies React's intuitive UI design approach to both iOS and Android, allowing developers to utilize native UI elements while maintaining complete access to the platform's capabilities.The use of declarative views makes the code more predictable and simplifies debugging. Moreover, its portability enables code reuse across various platforms, including iOS, Android, and more, streamlining the development process.

2.8.3. Expo

Expo is a production-grade React Native Framework. Expo provides developer tooling that makes developing apps easier, such as file-based routing, a standard library of native modules, and much more.

Expo's Framework is free and open source, with an active community on GitHub and Discord. The Expo team works in close collaboration with the React Native team at Meta to bring the latest React Native features to the Expo SDK.

The team at Expo also provides Expo Application Services ,EAS , an optional set of services that complements Expo, the Framework, in each step of the development process.

2.8.4. React Navigation

React Navigation is a comprehensive library for navigating between different screens within a React Native application. It emerged as a semi-official solution for handling navigation after the built-in navigation API was phased out from React Native.

React Navigation is designed to tackle several challenges associated with navigation in mobile applications:

- **Platform-Specific Navigation:** Navigation mechanisms significantly differ between iOS and Android platforms. iOS utilizes view controllers, while Android employs activities. These platform-specific APIs not only function differently but also present distinct user interfaces. React Navigation aims to support the unique look and feel of each platform while maintaining a consistent JavaScript API.
- **Mapping Navigation APIs to Views:** Native navigation APIs do not have a direct correspondence with "views". Components in React Native, such as View, Text, and Image, roughly map to an underlying native "view". However, there isn't a direct equivalent for some of the navigation APIs, making it challenging to expose these APIs to JavaScript.

- **Stateful Navigation:** Unlike web navigation, which is typically stateless (a URL or route directs a user to a single screen/page), mobile navigation is stateful. The user's navigation history is preserved in the application, allowing the user to return to previous screens. A stack of screens can even include the same screen multiple times.

React Navigation provides a sufficiently configurable solution for most applications, effectively addressing these challenges. It offers a range of features, including stack navigation, tab navigation, drawer navigation, and more. It also supports deep linking, which allows specific screens in your app to be opened via a URL.

In academic terms, React Navigation can be seen as a critical component in the architecture of React Native applications, providing a flexible and efficient solution for handling navigation and routing. Its design principles align with the overall philosophy of React Native, offering a single, consistent JavaScript API that abstracts away the complexities of platform-specific navigation [7].

2.8.5. Axios

Axios is a promise-based HTTP client for JavaScript used to make HTTP requests from both the browser and Node.js. It plays a vital role in handling network requests and responses in mobile app development. Axios simplifies asynchronous operations, making it easier to manage data flows and error handling.

2.9. User Interface and User Experience (UI/UX) Design

User experience plays a critical role in the success of a mobile application. This section explores essential UI/UX design principles and the methodologies used to create an intuitive and engaging user interface.

2.9.1. Design Principles

Effective UI/UX design is based on principles such as:

- **Simplicity**: Ensuring the interface is easy to use and understand.
- **Consistency**: Maintaining uniformity in design elements and interactions across the app.
- **Accessibility**: Making the app usable for people with varying abilities and disabilities.

2.9.2. Responsive Design

Responsive design ensures that the application looks and works well on different screen sizes and orientations. Techniques such as Flexbox and CSS Grid Layout are commonly used to achieve responsiveness. Ensuring a responsive design enhances the user experience by providing a consistent look and feel across various devices.

2.10. Development Environment

IDE: Integrated Development Environment The Integrated Development Environment, or IDE, helps programmers to combine the various parts of building a computer software into one. By consolidating common software writing tasks into a single application;editing source code, creating executables, and debuggin.IDEs boost programmer productivity[8].

2.10.1. VSC:Visual Studio Code

Visual Studio Code is a streamlined yet powerful source code editor that operates on Computer desktop and supports Windows, macOS, and Linux. It comes with integrated support for JavaScript, TypeScript, and Node.js, and boasts a strong ecosystem of extensions for various languages and runtimes, including C++, C#, Java, Python, PHP, Go, and .NET[9].

2.10.2. Version Control Systems

Version control, also known as **versioning** or **source control**, is the practice of managing changes to source code. It ensures that changes are both trackable and reversible.

Types of Version Control There are three main forms of version control systems:

1. **Local Version Control**: Changes are stored locally before being pushed to a single version of code in a database.
2. **Central Version Control**: Hosts different versions of the code in a centralized repository.
3. **Distributed Version Control**: Each local repository fully mirrors the central repository, including its history.

Benefits of Version Control in Project Management Version control offers several benefits:

- Maintains the latest version of all files.
- Preserves a history of changes.
- Facilitates collaboration.
- Prevents data loss.
- Increases transparency and accountability.

2.11. Summary

This chapter reviewed the key theoretical concepts and technologies that form the foundation of our mobile application development project. By understanding the principles of mobile app development, backend development, networking, UI/UX design, and existing literature, we can effectively address the challenges and leverage best practices in our implementation.

Chapter 3

System Design and Implementation

3.1. Introduction

In this chapter, we delve into the intricate system design and implementation aspects of our final year project. The backend infrastructure serves as the foundational core, encompassing data management, authentication protocols, and essential business logic. On the other hand, the frontend interface represents the user-centric interaction layer where users engage with the application's features.

3.2. System Overview

3.2.1. Architecture Diagram

The architectural representation of our application encapsulates the critical components orchestrating its functionality. Let's dissect the architectural diagram illustrating the system's structural framework and operational interplay in fig 3.1:

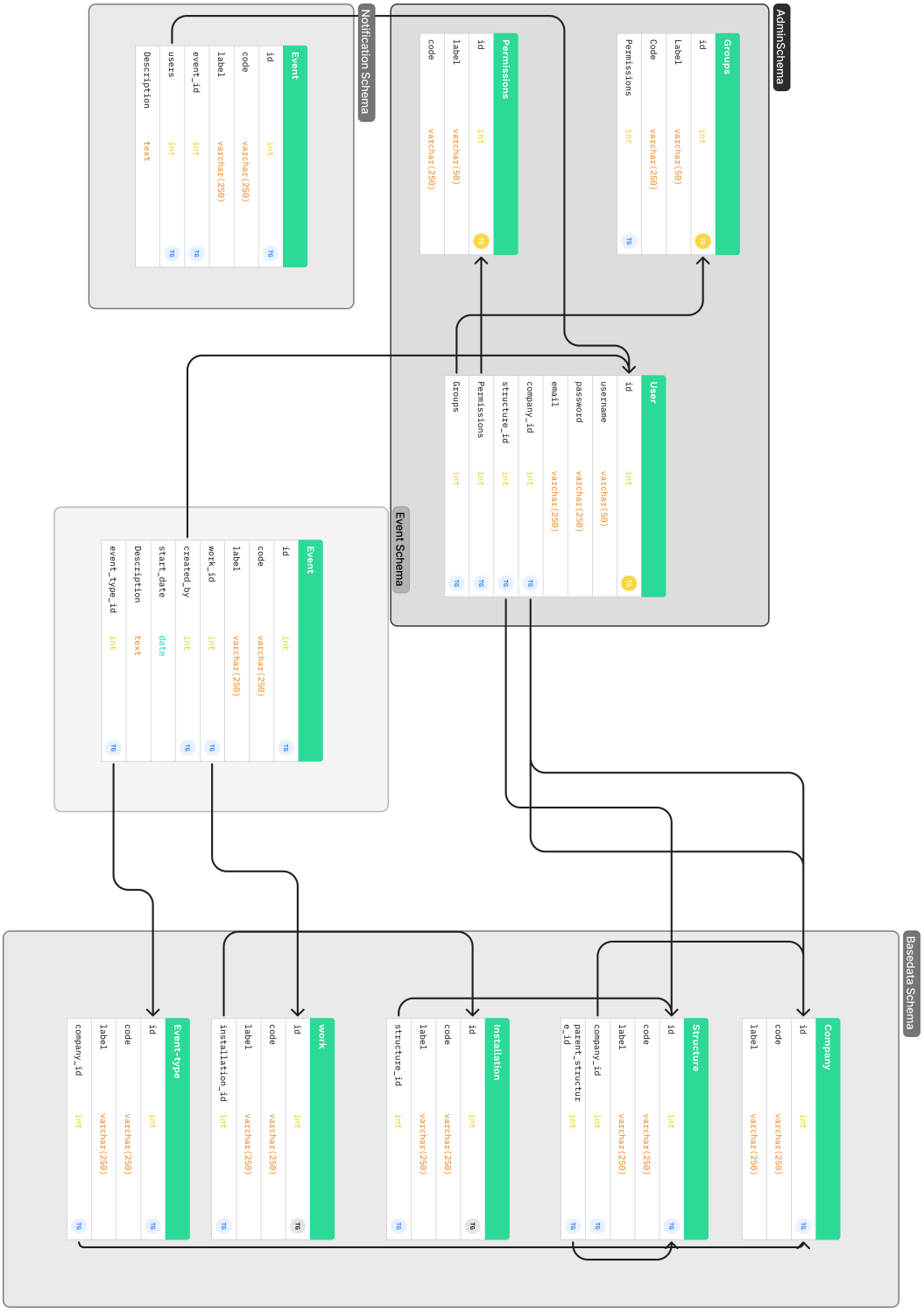


Fig. 3.1. Application System database schemas

Back-end: The backend system comprises interconnected data tables forming a robust operational backbone, with each table's reliance on others establishing

critical dependencies.
Front-end: The frontend interface delineates into two primary segments:

- Non-authenticated user screen (login interface).
- Authenticated user screen. The integration and collaboration between these interfaces are elucidated in the visual representation provided:

3.2.2. System Components

The system architecture is composed of two pivotal entities: the backend infrastructure and the frontend interface, housing an array of components crucial for seamless system operation. These components will be elaborated upon extensively in subsequent sections.

3.2.3. Requirements Specification

- **Performance:** Optimal loading speed target of 2 seconds on standard broadband connections.
- **Usability:** Prioritization of intuitive and user-friendly interface design for enhanced accessibility.
- **Security:** Robust encryption mechanisms for safeguarding sensitive data in transit and at rest, coupled with stringent role-based access control measures.
- **Compatibility:** Ensuring seamless operation across prevalent platforms such as iOS and Android devices.
- **Maintainability:** Emphasis on modular code architecture and comprehensive documentation to facilitate ease of maintenance and scalability.

3.3. Back-end Design and Implementation

Database Design:

The database schema, as portrayed in **Figure 3.1**, encompasses four crucial schemas:

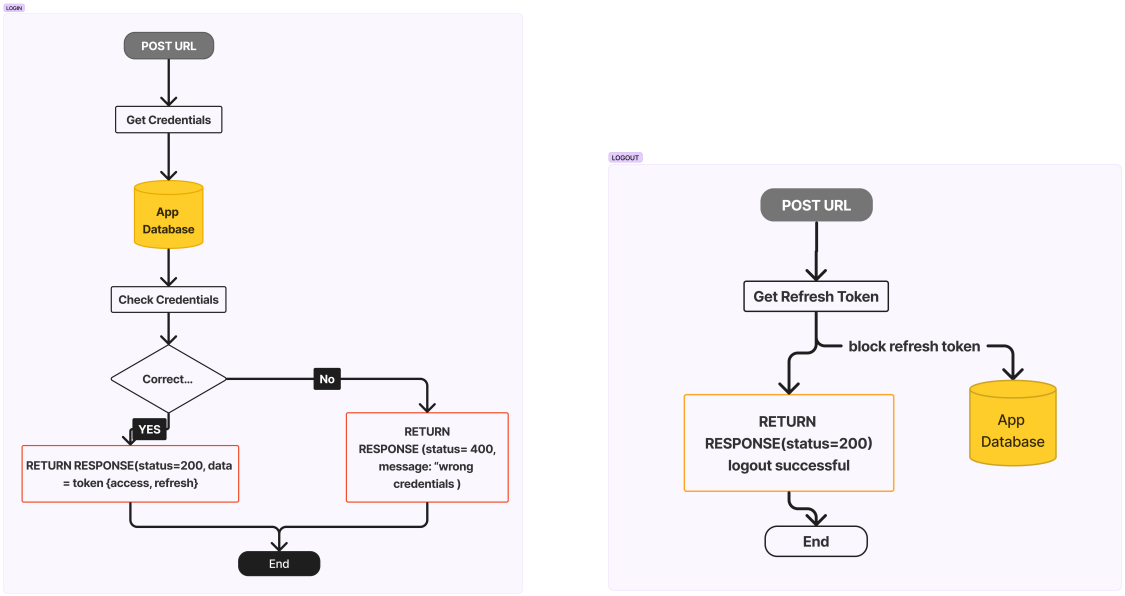
- **Admin Schema:** Repository for user profiles, permissions configuration, and group associations.
- **BaseData Schema:** Central repository defining the company model structure with intricate hierarchical relationships (e.g., installations and work processes).
- **Event Schema:** Segregated schema dedicated to storing pivotal event-related data crucial for core application functionality.

- **Notification Schema:** Specific segment allocated for managing notifications and directing them to designated users.

API Design:

The API architecture within the application ecosystem is categorized into three distinct domains:

- **Login/logout APIs:**
  login and logout logic is illustrated in flowcharts fig 3.2 (a), and (b) respectively.
  Login Mechanism: User authentication process initiated through credential submission at a designated endpoint, followed by server-side validation and token issuance.
  Logout Functionality: Secure logout mechanism involving token invalidation and confirmation of successful session termination.



(a) Login API Endpoint

(b) Logout API Endpoint

Fig. 3.2. Authentication APIs Workflow

- **GET APIs:** GET APIs follow a standardized procedure where users submit requests with access tokens in the headers. The API responds by providing pertinent information based on user roles and permissions. An exemplar of fetching event data is presented below:

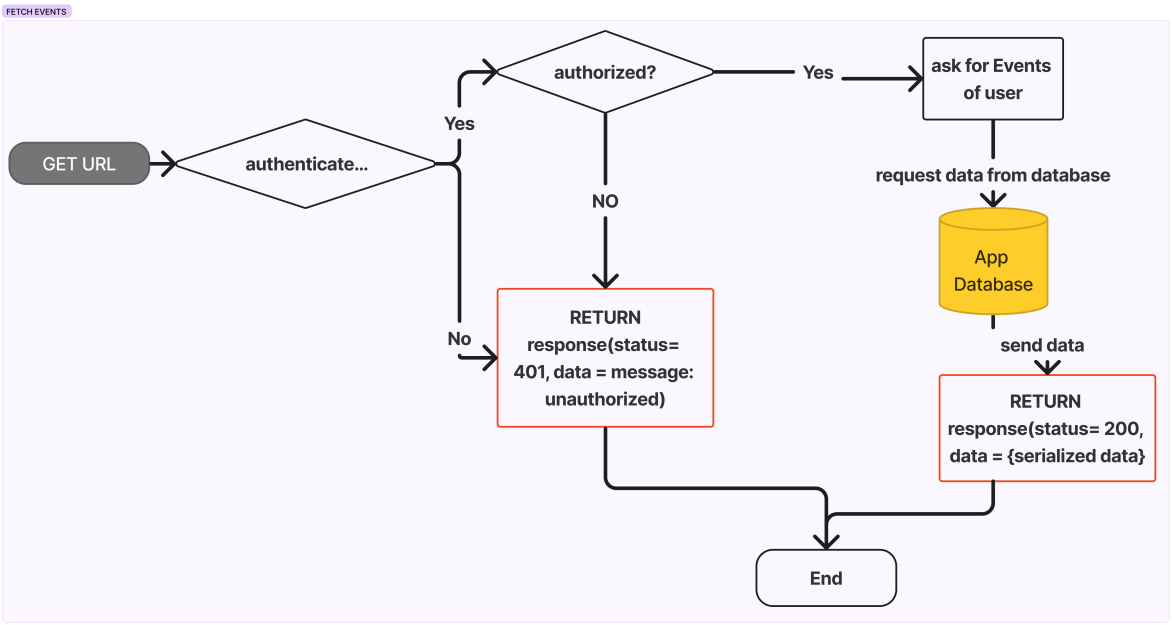


Fig. 3.3. Fetch Events API Endpoint

- **Post APIs:** Post APIs adhere to a consistent methodology akin to GET APIs, focusing on adding data objects to the database. Users transmit POST requests with requisite data, which undergo validation based on permissions and data integrity. Successful data addition prompts a response with the saved data, while errors are communicated for resolution.

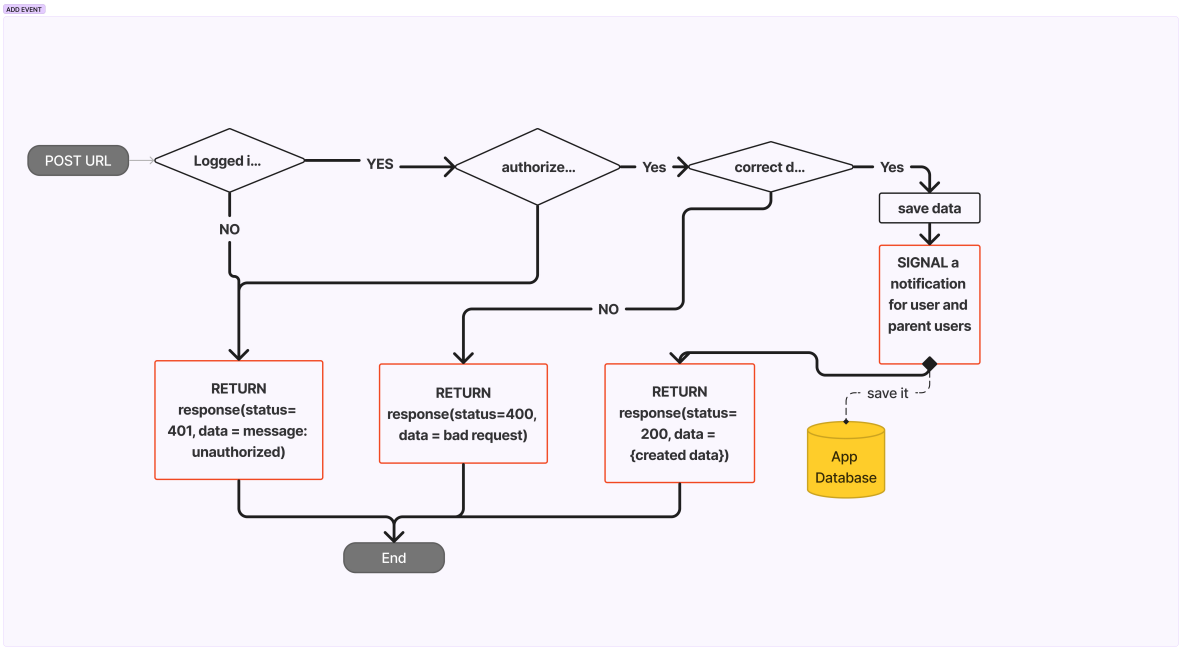


Fig. 3.4. Add Event API Endpoint

Each API category encompasses distinctive endpoints tailored to optimize user interaction and streamline system operations effectively. This meticulous API de-

sign ensures a robust and efficient user experience within the application ecosystem, aligning with stringent technical standards and operational requirements.

3.4. Front-end Design and Implementation

the front-end is the most important part for users as they interact with it and all the operations are done through it, so our design is focussing on implementing a user friendly application, easy to interact with and can handle all operations needed.

3.4.1. Front-end System Design

System Design

As illustrated in Fig 3.5, our front-end system operates on the following logic: initially, when users enter the application, it checks for any previously saved token and attempts to update it. If the update is successful, the user is granted access to the protected screens and begins fetching notifications if any new notification is detected it will be displayed in the status bar of the phone as shown in fig 3.6. Otherwise, the user is redirected to the login screen.

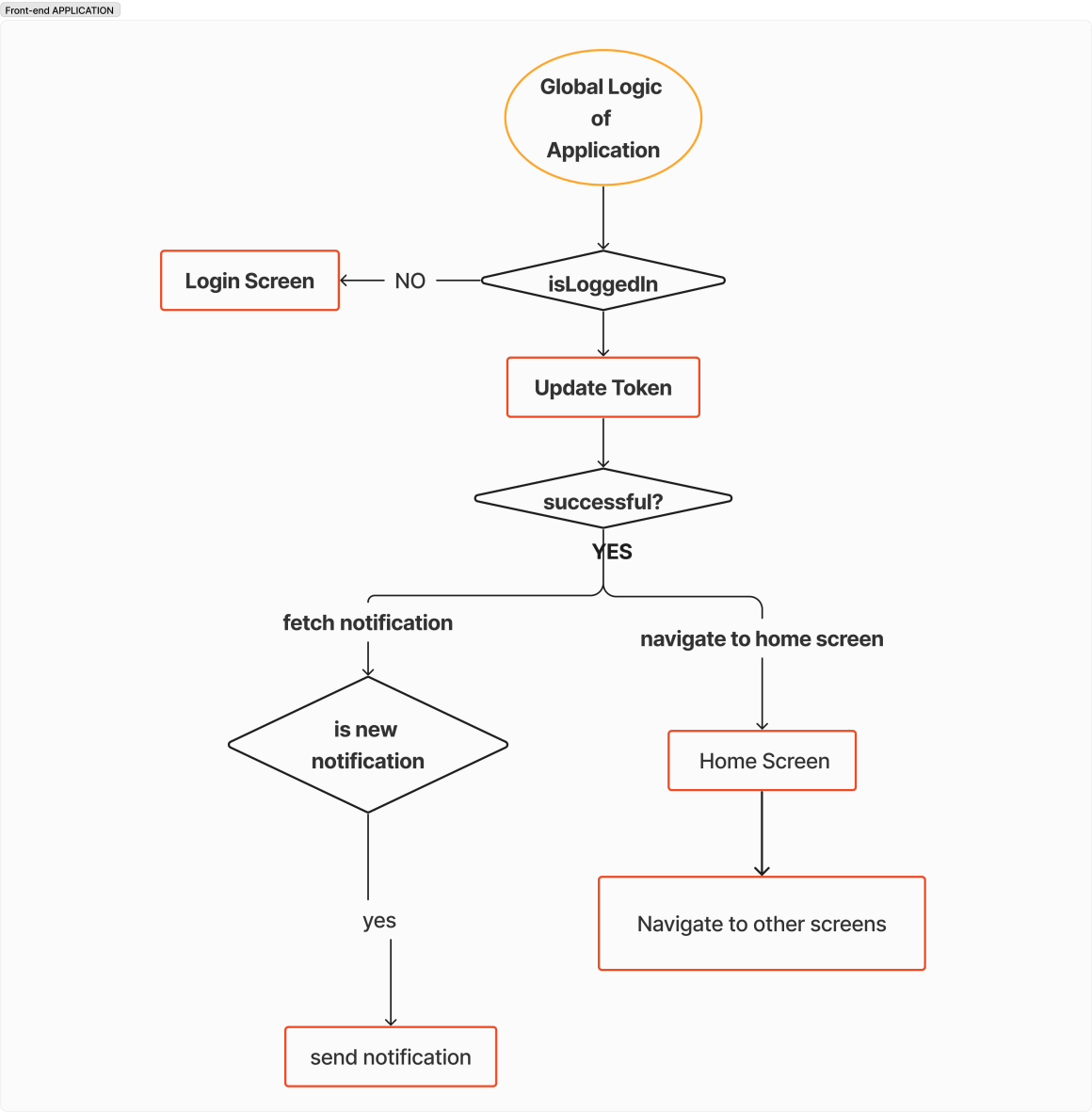


Fig. 3.5. User-Oriented Application Front-end Layout

As illustrated in Fig 3.5, our front-end system operates on the following logic: initially, when users enter the application, it checks for any previously saved token and attempts to update it. If the update is successful, the user is granted access to the protected screens and begins fetching notifications if any new notification is detected it will be displayed in the status bar of the phone as shown in figure bellow. Otherwise, the user is redirected to the login screen.

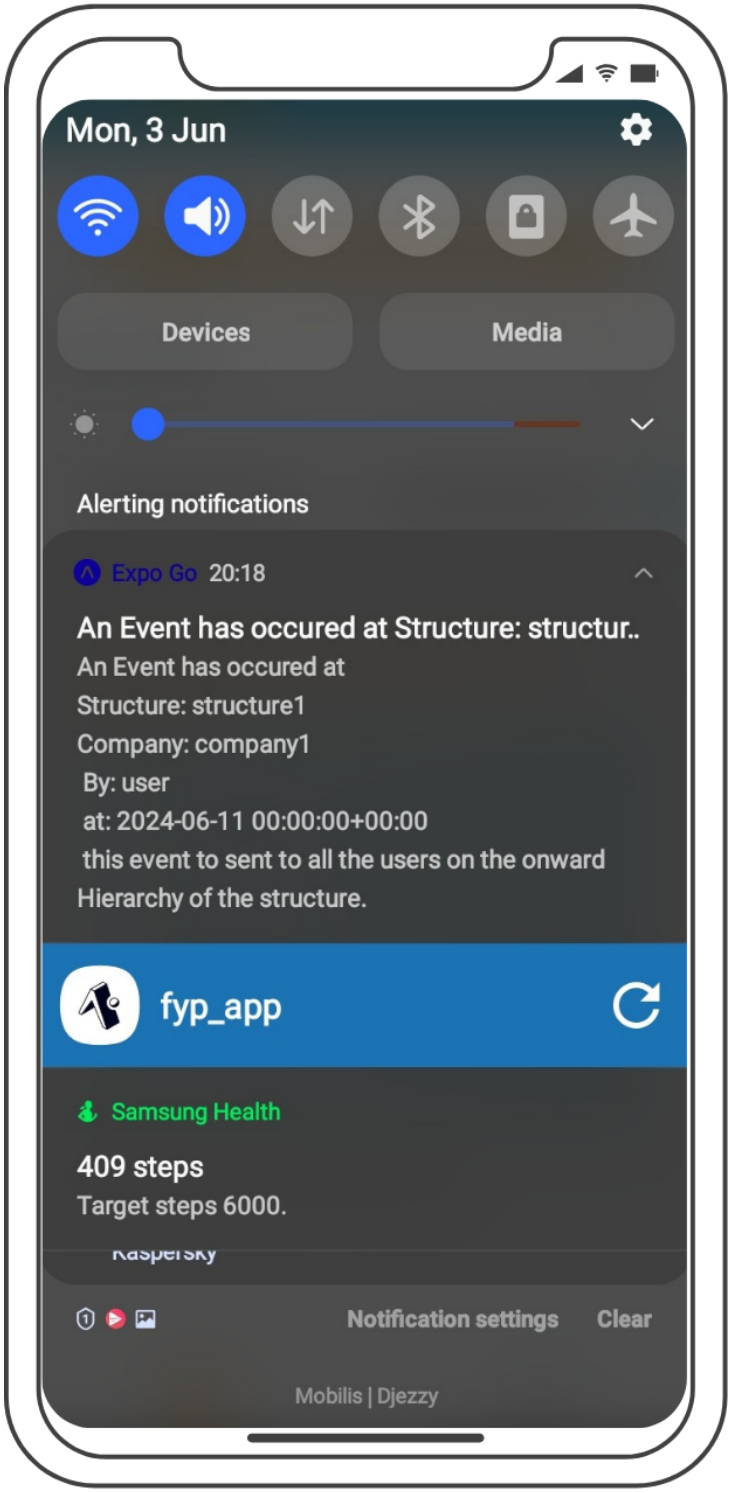


Fig. 3.6. notification received

Protected screens include: Home screen, Notification screen, Event screen, and Change Password screen. Users can navigate between the Notification screen and Home screen via the bottom tab bar, and between other screens (Home, Event, and Change Password) via the Drawer.

Design of Each Screen:

- **Home Screen:**

  The application retrieves the access token and sends a GET request to display statistics of events created by the user. Users can select child structures and specify a start and end date to obtain statistics for each structure within the selected date range.

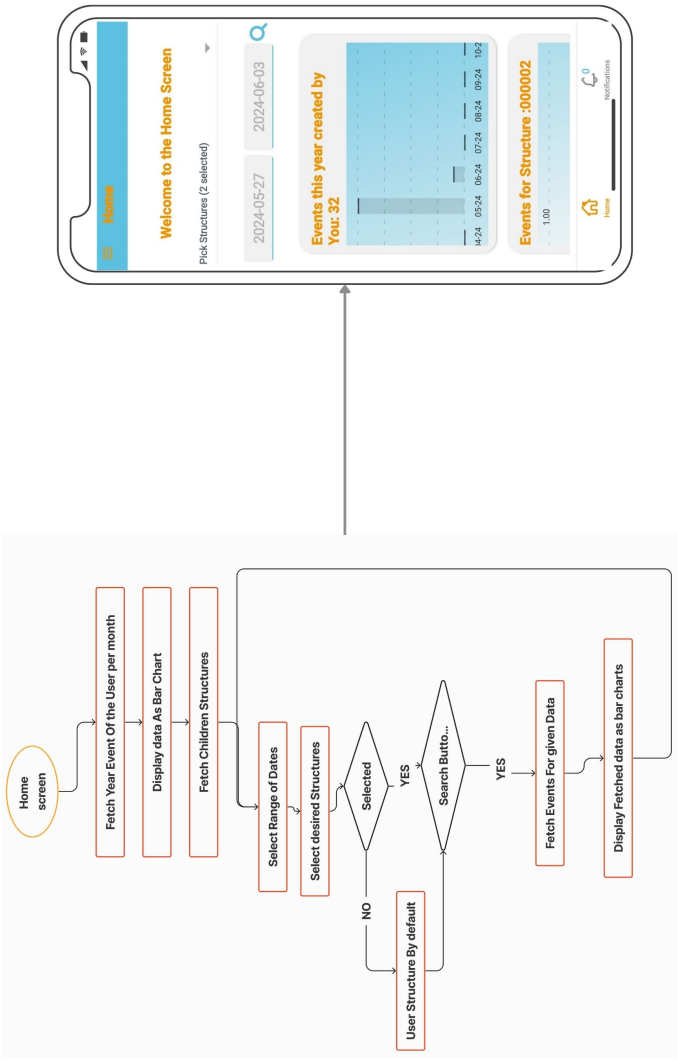


Fig. 3.7. HomeScreen layout

- **Notification Screen:** Upon navigating to notifications, the app sends a GET request. After receiving the response, notification cards are displayed with title, description, and read/unread status. as is fig 3.8

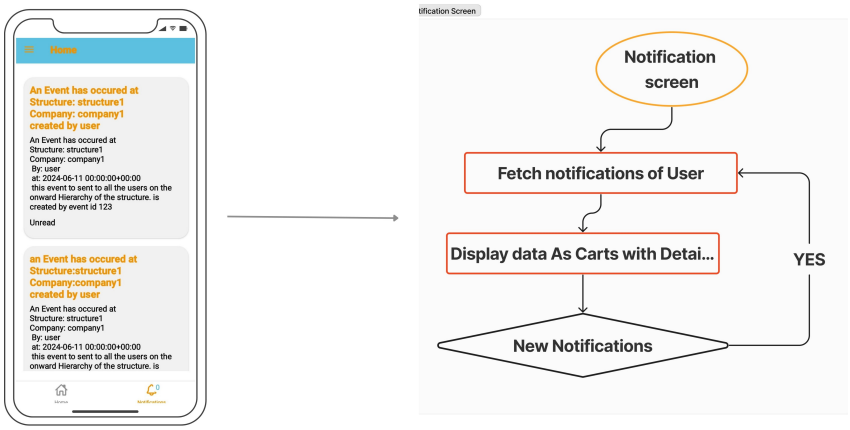


Fig. 3.8. Notification Screen Layout

- **Event Screen:** When navigating to the Events screen, users see a plus button to declare a new event. Below it, cards displaying events created by the user are shown. this can be seen in fig 3.9

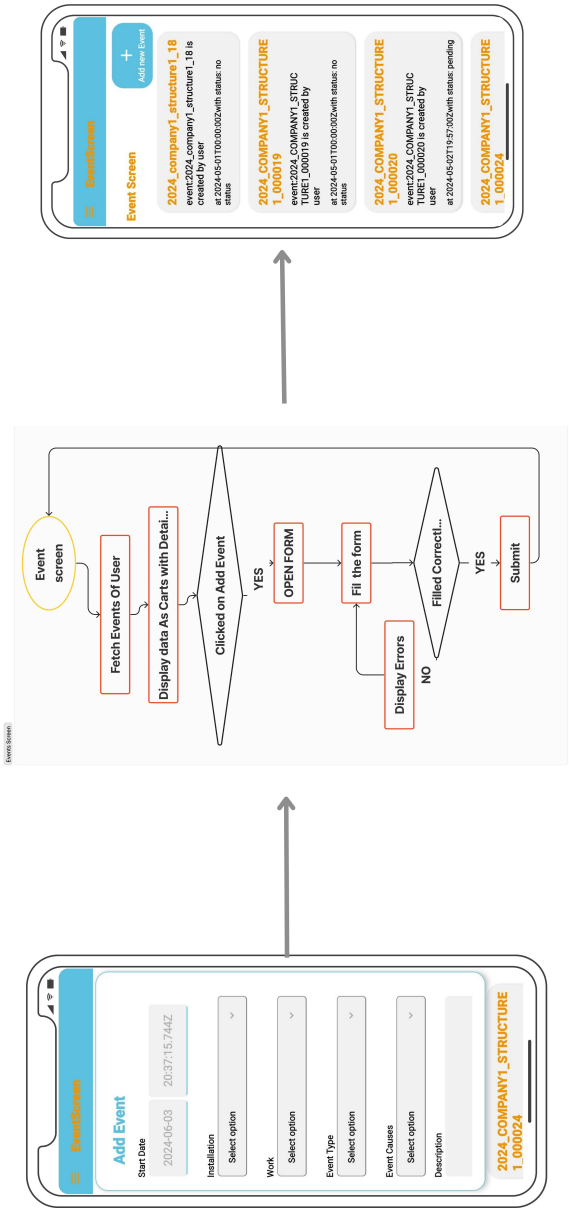


Fig. 3.9. event screen Layout

- **Change Password Screen:** This screen includes three input fields: current password, new password, and confirm new password. Users must fill all fields with the required specifications to send a PATCH request to change the password. If the request is successful, the user is logged out and must reconnect with the new password. The Change password screen is shown in fig 3.10.

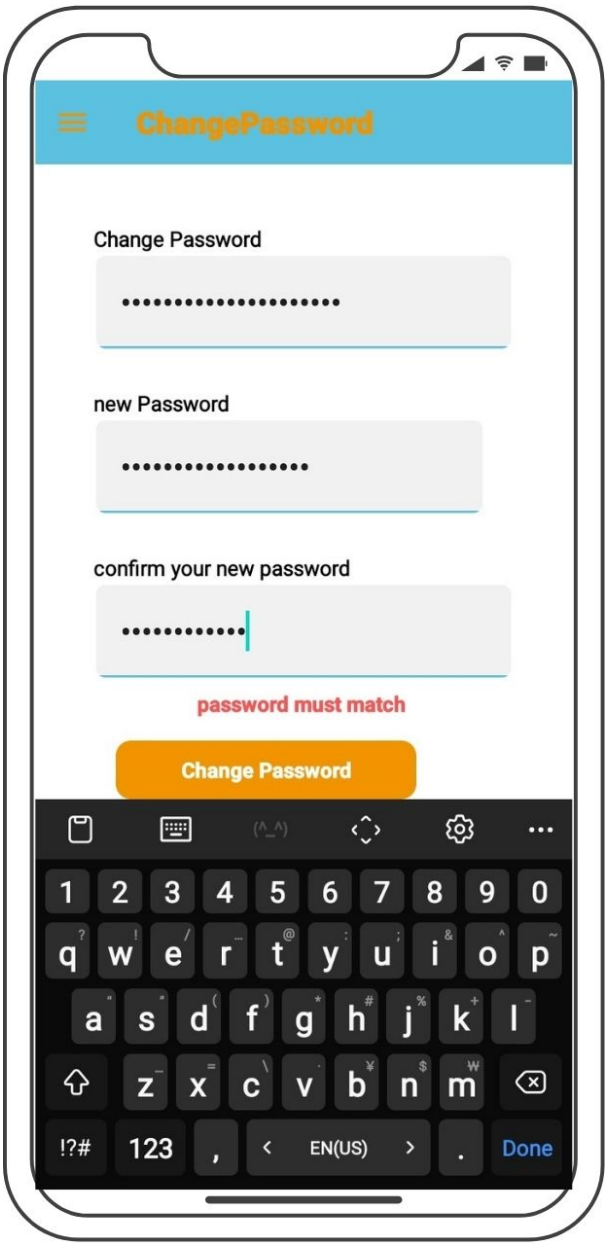


Fig. 3.10. Change Password Screen

Design Principles

The front-end design adheres to principles of consistency, usability, and scalability. Consistent design elements are used throughout the interface to ensure a cohesive user experience. Usability is prioritized through intuitive navigation and clear information hierarchy. Scalability is considered by designing components that can easily accommodate future feature additions without major overhauls.

3.5. Challenges and Solutions

Challenges we faced included dealing with Axios network issues during application development. These issues occurred when the back-end server was running on localhost with the IP address `127.0.0.1`, making it inaccessible to the mobile device. To resolve this, we used the IP address of the local network and connected the phone to the same network source. Alternatively, we explored using tunneling services that run the server on the internet, providing an accessible URL address.

Efficiently handling asynchronous data flow was another challenge, which we addressed using asynchronous functions along with the `.then()` method. Additionally, we ensured a consistent design across different browsers and devices by utilizing responsive design techniques such as Flexbox, maintaining a consistent layout across various devices.

3.6. Future Enhancements

3.6.1. Improvements

Future enhancement include adding more features to the application such as:

- **User Profiles:**Introducing personalized user accounts where individuals can manage their information, preferences, and settings. This could include profile pictures, contact details, and customizable options to tailor the app experience to each user's needs.
- **Animations:**Implementing smooth and visually appealing animations to enhance the user interface. This could involve transitions between screens, animated buttons, and feedback for user interactions to make the app more engaging and interactive.
- **More Security Features:** adding More security to the application with the One Time Password which can add a layer of security insurance

3.7. Results

Our application has successfully met the needs of the company, as it can send and receive events seamlessly, ensuring real-time communication and efficient data handling. The implementation of this capability has significantly improved the workflow by allowing instant updates and notifications about operational interruptions, which are critical for timely decision-making. Additionally, the application's robust event management system integrates smoothly with existing in-

frastructure, minimizing downtime and enhancing overall productivity. This has not only streamlined internal processes but also bolstered the company's ability to respond swiftly to incidents, thereby maintaining high service reliability and customer satisfaction. The positive feedback from users further validates the application's effectiveness and underscores its vital role in supporting the company's operational goals.

3.8. Conclusion

In this chapter, we have explored all aspects of the application system, including the front-end, back-end, and design principles. In the next chapter, we will provide the conclusion.

Chapter 4

Conclusion

In conclusion, the development of the OICI mobile application marks a significant milestone in the Sonelgaz Group's ongoing efforts to enhance operational efficiency and improve service delivery. By leveraging advanced mobile technologies, the OICI system has successfully centralized critical data and streamlined communication processes, ensuring that operational interruptions are managed with greater precision and responsiveness. This project not only underscores the importance of technological innovation within the energy sector but also exemplifies Sonelgaz's dedication to maintaining high standards of reliability and customer satisfaction.

The comprehensive exploration of the project's theoretical and practical facets, as documented in this report, highlights the collaborative efforts and strategic planning that were pivotal to its success. From understanding the intricate needs of the Sonelgaz Group to meticulously designing and implementing the application, each step has been driven by a commitment to excellence and innovation.

Looking forward, the OICI system is poised for further enhancements, which will undoubtedly bolster its capabilities and user experience. The insights and experiences gained from this project provide a strong foundation for future initiatives aimed at optimizing the operational efficiency of the Sonelgaz Group.

Ultimately, the successful implementation of the OICI application stands as a testament to the transformative power of technology in addressing complex operational challenges and underscores the Sonelgaz Group's unwavering commitment to progress and customer-centric service delivery.

References

[1] c. Meta. “What is backend development”. Accessed: 2024-05-30. (2024), [Online]. Available: https://www.coursera.org/articles/back-end-developer.

[2] postegreSQL. “Postgresql”. Accessed: 2024-06-03. (2024), [Online]. Available: https://www.postgresql.org/docs/current/intro-whatis.html.

[3] F5. “What is an api”. Accessed: 2024-05-30. (2024), [Online]. Available: https://www.f5.com/glossary/web-app-and-api-protection-waap.

[4] mdn web docs. “An overview of http”. Accessed: 2024-05-30. (2024), [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.

[5] c. Meta. “Http methods”. Accessed: 2024-05-30. (2024), [Online]. Available: https://www.coursera.org/learn/meta-working-with-data/supplement/REJZP/http-methods-status-codes-and-response-types.

[6] Facebook. “What is react”. Accessed: 2024-05-30. (2024), [Online]. Available: https://github.com/facebook/react/blob/main/README.md.

[7] R. N. Express. “React navigation”. Accessed: 2024-05-30. (2024), [Online]. Available: https://www.reactnative.express/app/navigation.

[8] C. Team. “What is an ide”. Accessed: 2024-05-29. (May 2024), [Online]. Available: https://www.codecademy.com/article/what-is-an-ide.

[9] Microsoft. “What is vscode”. Accessed: 2024-05-29. (May 2024), [Online]. Available: https://visualstudio.microsoft.com/#vscode-section.