



RV Educational Institutions[®]
RV College of Engineering[®]

Autonomous
Institution Affiliated
to Visvesvaraya
Technological
University, Belagavi

Approved by AICTE,
New Delhi

Go, change the world

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

App Development

SUMMER INTERNSHIP REPORT

(21AII310)

Submitted by

Dev

1RV21AI017

Under the guidance of

Akash Sharma

Orizzel

2023-2024

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Bengaluru– 560059



CERTIFICATE

Certified that the Internship work titled '**App Development**' is carried out by **Dev 1RV21AI017** in partial fulfilment for the requirement of degree of **Bachelor of Engineering in Artificial Intelligence and Machine Learning** of the Visvesvaraya Technological University, Belagavi during the year 2022-2023. Internship work report has been approved as it satisfies the academic requirements in respect of the work prescribed by the institution for the said degree.

Dr. Vijayalakshmi M N
Coordinator-Internship
Department of AI & ML
RVCE, Bengaluru

Dr. B. Sathish Babu
Professor and HoD
Department of AI & ML
RVCE, Bengaluru

External Viva

Name of Examiners

Signature with Date

1.

2.

CERTIFICATE

ORIZEL



CERTIFICATE OF APPRECIATION

This is to certify that

Dev Juneja

A student of B.Tech Artificial Intelligence and machine from 'R V College of Engineering, Bengaluru has completed his training from 13 Nov to 13 Dec at Orizzel India Pvt Ltd.

During the tenure of his internship he was found to be very hardworking , sincere and dedicated . He has attained a good knowledge & on job training . He has performed as per the expectation during his internship. We wish him the best for his future.

Mr. Jatin Sharma
CEO , Orizzel

Internship Summary

During my internship at Orizzel, I made significant contributions to a project focused on developing a food delivery application using Kotlin and Firebase. My responsibilities included implementing key features such as user authentication, database management, and server-side logic.

Throughout the internship, I actively engaged in learning and applying my knowledge in Kotlin and Firebase to real-world projects. I effectively managed my time, ensuring tasks were completed within set deadlines and contributing to the overall project progress.

One of my major contributions was in designing and implementing user authentication and authorization systems, ensuring the security and integrity of the food delivery platform. Additionally, I played a key role in structuring the database schemas and implementing CRUD operations to effectively manage restaurant and user data.

Collaboration was integral to the project's success, and I actively participated in team meetings, shared ideas, and provided support to my colleagues. Effective communication and teamwork enabled us to overcome challenges and achieve project milestones.

Throughout the internship, I demonstrated adaptability and problem-solving skills, successfully navigating through technical challenges and finding innovative solutions. My ability to quickly adapt to changing project requirements and effectively utilize available resources contributed to the project's success.

Overall, my internship at Orizzel provided me with valuable hands-on experience in developing applications using Kotlin and Firebase, enhancing my technical skills and preparing me for future roles in software development.

Table of Contents

Chapter 1		
Profile of the Organization		Page Number
1.1	Organizational Structure	6
1.2	Products	6
1.3	Services	7
1.4	Business Partners	8
1.5	Financials	8
1.6	Manpower	9
1.7	Societal Concerns	9
Chapter 2		
Activities of the Department		
2.1	Specific functions	10
2.2	Techniques adopted	10
2.3	Tools used	11
2.4	R&D activities	12
Chapter 3		
Tasks Performed		
3.1	Tasks performed : Technical	13
3.2	Tasks performed : Non-Technical	20
Chapter 4		
Reflections		
4.1	Technical Knowledge acquired	22
References		23

1.Profile of the Organization

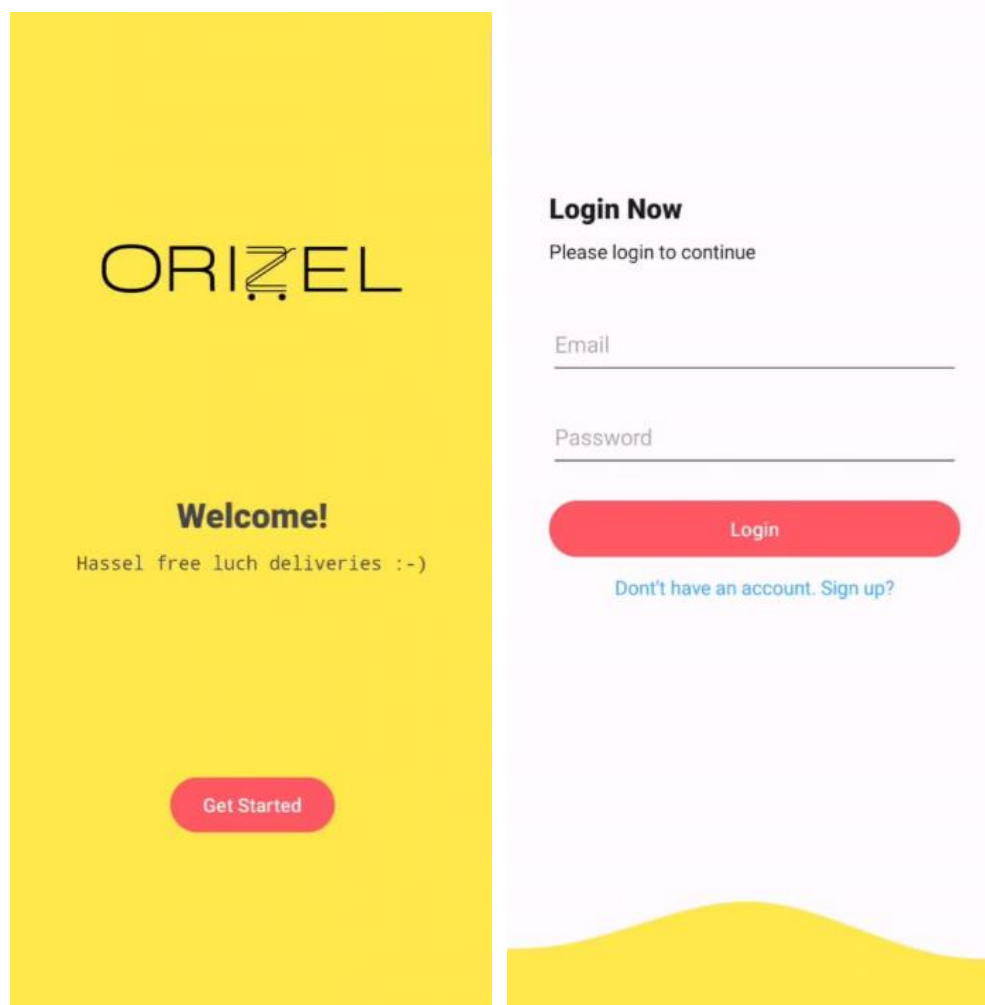
1.1 Organizational Structure

Orizzel, a dynamic food tech company specializing in on-campus food delivery services, has emerged as a leader in the industry. With a team comprising approximately 30-40 dedicated employees, Orizzel has successfully crafted a seamless user experience through the development of an integrated app, chatbot, and website. Meanwhile, a notable player in the broader realm of IT services, the other company is recognized for its expertise in AI chatbots, staffing, and IoT solutions. Their proficiency spans across web, mobile, and enterprise domains, positioning them as one of the top AI consultants. Together, these companies exemplify diverse facets of the evolving technological landscape, showcasing innovation in both specialized food delivery solutions and comprehensive IT services

1.2 Products

Orizzel App

Orizzel, the innovative food delivery app, seamlessly connects users with a diverse range of culinary delights, delivering convenience and flavor to their doorstep.


















ChatBot

Introducing Orizzel's cutting-edge Food Ordering Chatbot on WhatsApp, where the future of culinary convenience meets seamless interaction. This intuitive chatbot transforms the traditional food ordering experience into a personalized and efficient process. Customers can simply share their food preferences, and the chatbot, armed with artificial intelligence, guides them through a diverse menu to make the perfect selection. Location details can be effortlessly communicated, ensuring a swift and accurate delivery to the customer's doorstep. The chatbot also caters to special requests and offers personalized recommendations, making the entire ordering journey delightful. With exclusive discounts and promotions available through chat interactions, Orizzel's WhatsApp Chatbot brings innovation and ease to every meal, promising a culinary adventure with every conversation.

1.3 Services

1. Efficient In-Campus Delivery
2. Diverse Culinary Options
3. Customizable Meal Plans
4. Real-Time Order Tracking
5. Exclusive Campus Discounts
6. Contactless Payments
7. Personalized Recommendations
8. Environmental Sustainability
9. 24/7 Customer Support
10. Community Engagement Events

1.4 Business Partner

1.5 Manpower

Employee Data

- Orizzel has 40 Employees.
- Orizzel grew their employee count by 46% last year

1.6 Societal Concerns

1. **Health and Nutrition:** Ensuring that the food provided is nutritious and meets certain health standards is crucial. Many students rely on campus food delivery services for their meals, and promoting healthy eating habits can have a positive impact on their well-being.
2. **Environmental Impact:** Minimizing the environmental footprint associated with food delivery operations is important. This includes reducing single-use packaging, promoting eco-friendly delivery methods, and sourcing ingredients sustainably.
3. **Safety:** Ensuring the safety of both delivery personnel and customers is paramount. This involves implementing strict hygiene protocols, providing proper training to delivery staff, and adhering to food safety regulations.
4. **Affordability and Accessibility:** Making sure that food delivery services remain affordable and accessible to all students, including those from diverse socioeconomic backgrounds, is essential. Offering discounts, meal plans, or subsidies can help address affordability concerns.
5. **Community Impact:** Supporting the local community and fostering positive relationships with campus residents and businesses is important. This could involve sourcing ingredients locally, partnering with campus organizations for events or initiatives, and contributing to local causes.

1.7 Professional practices

Orizzel, the food delivery company within the student campus, upholds professional practices by prioritizing health and nutrition standards in its menu offerings. Environmental sustainability is central to its operations, achieved through eco-friendly delivery methods and local ingredient sourcing. The company maintains stringent safety measures to ensure the well-being of both customers and delivery staff. Affordability and accessibility are key considerations, with Orizzel offering discounted meal plans and subsidies to cater to diverse student needs. Community engagement is fostered through partnerships with local businesses and support for campus events. Data privacy is a top priority, with Orizzel implementing robust measures to safeguard customer information. Inclusivity is promoted through menu options accommodating various dietary preferences and restrictions. Efficient delivery routing helps mitigate traffic congestion on campus premises. Orizzel's commitment to professional practices sets a benchmark for excellence in campus food delivery. By adhering to these principles, Orizzel strives to enhance customer satisfaction and contribute positively to the campus community.

Activities of the Department

2.1 Specific Functions

Vision:

Become the most valued and trusted in-campus food delivery organization, delivering innovative solutions through deep collaboration with our partners.

Mission:

Orizzel strives to revolutionize the in-campus dining experience by providing seamless food delivery services through our app and WhatsApp bot. We offer innovative solutions tailored for campuses, catering to students and faculty. As trusted partners, we aim for long-term relationships, prioritizing customer satisfaction over short-term gains.

Values:

Our commitment lies in making campus dining a success by upholding values of integrity, honesty, empathetic leadership, and personal excellence. We continuously improve ourselves and foster mutual respect among our team and partners.

Services:

Orizzel is dedicated to providing in-campus food delivery services through our user-friendly app and WhatsApp bot, ensuring convenient and personalized meal options for students and faculty.

2.1 Techniques adopted



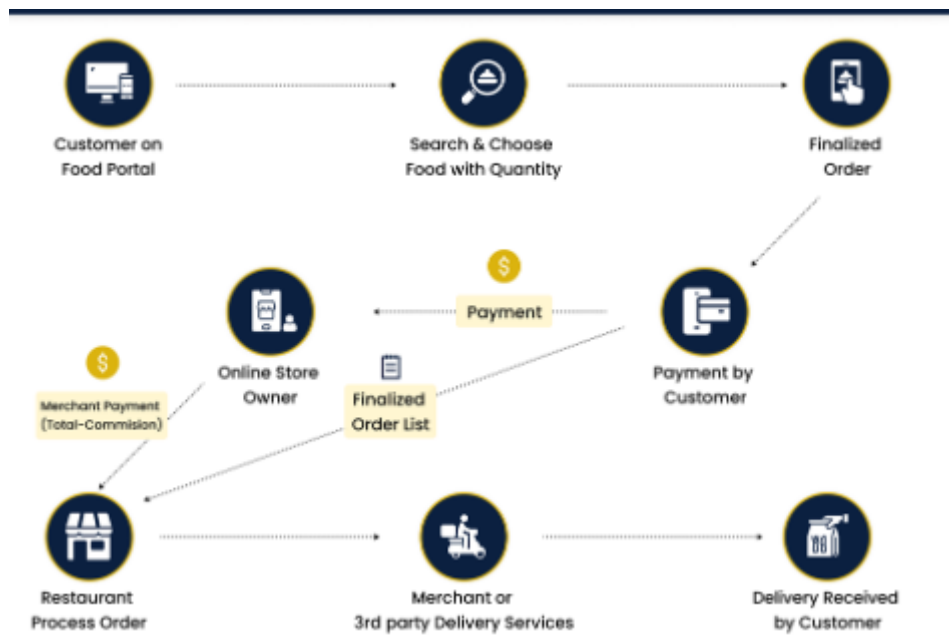
2.2 Tools Used

Orizzel, powered by a dynamic tech stack featuring Kotlin and Python, optimizes its in-campus food delivery app for innovation and efficiency. Robust databases such as MongoDB, PostgreSQL, and Elasticsearch ensure seamless data management, while React and Kotlin drive interactive user interfaces on web and mobile platforms. The app's cross-platform capabilities are enhanced by technologies like Ionic, extending accessibility. Real-time communication is facilitated by Socket.IO, and Docker ensures consistent app deployment. OpenAI and Cloud.AI drive artificial intelligence initiatives, while Fats API seamlessly integrates third-party services. AWS and Google Cloud Storage meet cloud computing needs, ensuring scalability and reliability. Orizzel's tech-forward approach, anchored by Kotlin and Python, enables the app to evolve and meet the dynamic demands of in-campus food delivery for the students there

Furthermore, Tailwind expedites flexible UI development, complementing the app's aesthetic appeal. The integration of advanced technologies like Milvus and Dgraph enhances data indexing and graph database functionalities, facilitating sophisticated data querying and analysis. The app's commitment to sustainability is evident through eco-friendly practices, including the use of biodegradable packaging. By harnessing the power of artificial intelligence and machine learning with OpenAI and Cloud.AI, Orizzel achieves intelligent automation and predictive analytics, refining the user experience. With AWS services and Google Cloud Storage ensuring secure and scalable cloud infrastructure, Orizzel's tech-savvy approach positions the app at the forefront of in-campus food delivery innovation.



2.3 R&D Activites



Tasks Performed

2.4 Technical

- a) Signup page -- This page contains Kotlin code for an Android activity implementing user signup functionality using Firebase Authentication, including input validation, email verification, and navigation to a login activity. It utilizes View Binding for efficient view access and Toast messages for user feedback.

```

app > src > main > java > com > orizel > activities > SignupActivity.kt
1  package com.orizel.activities
2
3  import android.content.Intent
4  import android.graphics.Color
5  import androidx.appcompat.app.AppCompatActivity
6  import android.os.Bundle
7  import android.widget.Toast
8  import com.google.firebase.auth.FirebaseAuth
9  import com.orizel.R
10 import com.orizel.databinding.ActivitySignupBinding
11
12 class SignupActivity : AppCompatActivity() {class foo {
13
14 }
15     private var binding: ActivitySignupBinding? = null
16     private lateinit var firebaseAuth: FirebaseAuth
17
18     override fun onCreate(savedInstanceState: Bundle?) {
19         super.onCreate(savedInstanceState)
20         binding = ActivitySignupBinding.inflate(layoutInflater)
21         setContentView(binding?.root)
22
23         //initializing firebase
24         firebaseAuth = FirebaseAuth.getInstance()
25
26         binding?.SignupBtn?.setOnClickListener {
27             signUpUser()
28         }
29         binding?.tvLogin?.setOnClickListener {

```

```

30             binding?.tvLogin?.setTextColor(Color.parseColor("#551A8B"))
31             intent = Intent(this, LoginActivity::class.java)
32             startActivity(intent)
33             finish()
34         }
35     }
36
37     //creating a user using firebase
38     private fun signUpUser() {
39         val email = binding?.etSemail?.text.toString()
40         val password = binding?.etSpassword?.text.toString()
41         val confirmPassword = binding?.etCnfrmPass?.text.toString()
42
43         //General ID and Password Authentication
44         if (email.isBlank() || password.isBlank() || confirmPassword.isBlank()) {
45             Toast.makeText(
46                 this, "Fields cannot be Blank",
47                 Toast.LENGTH_SHORT
48             ).show()
49             return
50         }
51         if (password != confirmPassword) {
52             Toast.makeText(
53                 this, "Passwords do not Match",
54                 Toast.LENGTH_SHORT
55             ).show()
56         }
57
58         //firebase create user functionality

```

```

        firebaseAuth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) {
                if (it.isSuccessful) {
                    val user = firebaseAuth.currentUser
                    user?.sendEmailVerification()
                        ?.addOnCompleteListener { emailTask ->
                            if (emailTask.isSuccessful) {
                                Toast.makeText(
                                    this, "Verification email sent",
                                    Toast.LENGTH_LONG
                                ).show()
                                binding?.etSemail?.text?.clear()
                                binding?.etSpassword?.text?.clear()
                                binding?.etCnfrmPass?.text?.clear()
                                firebaseAuth.signOut()
                                startActivity(Intent(this, LoginActivity::class.java))
                            } else {
                                Toast.makeText(this, "Failed to sent Verification email",
                                    Toast.LENGTH_LONG).show()
                            }
                        }
                }
            }
        } else {
            Toast.makeText(
                this, "Error Creating User",
                Toast.LENGTH_LONG
            ).show()
        }
    }
}

```

1. Package and Imports:

- b) Packages in Java/Kotlin are like containers for classes and other packages. They help in organizing code into logical groups.
- c) Imports are used to access classes, methods, and other members of packages that are not in the current package or sub-packages. They help in reducing verbosity by allowing the usage of short class names.

2. Class Declaration:

- d) ``SignupActivity`` is a subclass of ``AppCompatActivity``, which provides a framework to implement an activity in Android.
- e) ``AppCompatActivity`` is chosen for compatibility with older versions of Android as it provides features from the support library, including the action bar and other UI components.

3. Variables:

- f) ``binding`` is a variable used to access views from the layout XML file using ViewBinding. It's nullable because it's initialized later in the code.
- g) ``firebaseAuth`` is declared with ``lateinit`` indicating that it will be initialized later in the code. It's used to interact with Firebase Authentication services.

4. onCreate() Method:

- h) ``onCreate()`` is a lifecycle method called when the activity is first created.
- i) ``super.onCreate(savedInstanceState)`` calls the superclass implementation to ensure proper initialization of the activity.
- j) ViewBinding is used to inflate the layout defined in ``ActivitySignupBinding``, which

5. Initialization:

- `FirebaseAuth.getInstance()` retrieves the singleton instance of `FirebaseAuth`, which is used throughout the activity to manage user authentication.

6. Button Click Listeners:

- `setOnClickListener` is used to attach click listeners to UI elements. In this case, it's attached to the sign-up button and the login text view.
- When the sign-up button is clicked, `signUpUser()` function is invoked. When the login text view is clicked, it changes its text color and navigates to the login activity.

7. signUpUser() Function:

- This function encapsulates the logic for user signup process.
- It retrieves the email and password from input fields and validates them to ensure they are not empty and that the password matches the confirmation.
- It uses Firebase Authentication methods to create a new user account and send a verification email.
- Success and failure cases are handled with Toast messages to provide feedback to the user.

8. Firebase Operations:

- `createUserWithEmailAndPassword()` attempts to create a new user account with the provided email and password.
- `sendEmailVerification()` sends a verification email to the user's email address to confirm its validity.

9. Toast Messages:

- Toasts are brief messages displayed to the user to convey information or feedback. They are non-intrusive and appear at the bottom of the screen.
- In this code, Toasts are used to inform the user about the success or failure of user registration and email verification.

10. Navigation:

- After successful account creation and email verification, the user is navigated to the login activity using intents. Intents are a fundamental component of Android app navigation.

11. Resource Binding:

- `ViewBinding` simplifies the process of accessing views in the layout XML file by generating a binding class automatically. It helps in reducing boilerplate code and potential errors related to view references.

12. Error Handling:

- The code includes error handling mechanisms to deal with potential issues such as empty input fields, password mismatch, and failures during user creation or email verification. This ensures a smoother user experience by providing appropriate feedback in case of errors.

13. UI Interaction:

- Changing the text color of the "Login" text view when clicked provides visual feedback to the user, enhancing the interactivity of the user interface and improving the overall user experience.

b) Login Page -- The `LoginActivity` is an Android activity responsible for handling user authentication through Firebase. It provides functionality for users to log in using their email and password, with email verification checks and navigation to the signup page if needed.

```

1  package com.orizel.activities
2
3  import android.content.Intent
4  import android.graphics.Color
5  import androidx.appcompat.app.AppCompatActivity
6  import android.os.Bundle
7  import android.widget.Toast
8  import com.google.firebase.auth.FirebaseAuth
9  import com.google.firebase.auth.FirebaseUser
10 import com.orizel.R
11 import com.orizel.databinding.ActivityLoginBinding
12
13 class LoginActivity : AppCompatActivity() {
14
15     private var binding : ActivityLoginBinding? = null
16     private lateinit var firebaseAuth: FirebaseAuth
17     private var user : FirebaseUser? = null
18
19
20     override fun onCreate(savedInstanceState: Bundle?) {
21         super.onCreate(savedInstanceState)
22         binding = ActivityLoginBinding.inflate(layoutInflater)
23         setContentView(binding?.root)
24
25         firebaseAuth = FirebaseAuth.getInstance()
26
27         binding?.loginButton?.setOnClickListener {
28             login()
29         }

```

```

31         binding?.tvSignup?.setOnClickListener {
32             binding?.tvSignup?.setTextColor(Color.parseColor("#551A8B"))
33             intent = Intent(this, SignupActivity::class.java)
34             startActivity(intent)
35             finish()
36         }
37
38     }
39
40     //Here we are authenticating the user
41     private fun login(){
42         val email = binding?.etEmail?.text.toString()
43         val password = binding?.etPassword?.text.toString()
44
45         if(email.isBlank() || password.isBlank()){
46             Toast.makeText(this, "Fields cannot be Blank",
47                 Toast.LENGTH_SHORT).show()
48             return
49         }
50
51         //Login function of firebase
52         firebaseAuth.signInWithEmailAndPassword(email,password)
53             .addOnCompleteListener(this){
54                 if(it.isSuccessful){
55                     user = firebaseAuth.currentUser
56                     if(user!!.isEmailVerified) {
57                         Toast.makeText(this, "Login Successful",
58                             Toast.LENGTH_SHORT).show()

```

```

59                 intent = Intent(this, MainActivity::class.java)
60                 startActivity(intent)
61                 finish()
62             }else{
63                 Toast.makeText(this, "Please Verify your email",
64                     Toast.LENGTH_SHORT).show()
65             }
66         }
67         else{
68             Toast.makeText(this, "Authentication Failed",
69                 Toast.LENGTH_SHORT).show()
70         }
71     }
72
73 }
74
75 }

```


Aspects of the code:

1. Package and Imports:

- The package declaration provides a logical organizational structure for the codebase, aiding in maintainability and scalability.
- Imports are essential for accessing classes and functionalities required for Android app development and Firebase integration, promoting code reusability and readability.
- Specifically, importing classes related to view binding enhances efficiency in accessing UI elements, while Firebase Authentication imports facilitate user authentication tasks.

2. Class Declaration:

- Extending `AppCompatActivity` ensures compatibility with the Android support library, enabling the utilization of modern UI components and features.
- By defining the `LoginActivity` class, the code encapsulates login-related functionality within a modular and reusable component, adhering to object-oriented programming principles.

3. Properties:

- The `binding` property, initialized with an instance of `ActivityLoginBinding`, leverages view binding to seamlessly interact with UI elements defined in the layout XML file, promoting type safety and reducing boilerplate code.
- `firebaseAuth` facilitates authentication operations, such as user login and verification, by interfacing with Firebase Authentication services.
- The `user` property holds information about the currently authenticated user, enabling access to user-specific data and functionalities throughout the activity lifecycle.

4. onCreate() Method:

- Being a crucial lifecycle method, `onCreate()` orchestrates the initialization process of the activity, setting up essential components and UI elements.
- Utilizing view binding, the method inflates the layout and binds UI elements, streamlining UI development and maintenance efforts.
- Initialization of Firebase Authentication (`firebaseAuth`) ensures the availability of authentication services for user login and verification.
- Setting click listeners for UI elements, such as the login button and signup text view, establishes event-driven interaction patterns, enhancing user engagement and experience.

5. Login Button Click Listener:

- The login button click listener triggers the execution of the `login()` function, which encapsulates the logic for user authentication and navigation.

6. Signup Text View Click Listener:

- Upon clicking the signup text view, the text color changes to indicate user interaction feedback, enhancing the visual appeal and intuitiveness of the UI.
- Navigation to the `SignupActivity` via an intent facilitates seamless user flow between login and signup processes, enhancing user journey continuity.

7. login() Method:

- Responsible for handling user authentication, the `login()` method retrieves user credentials from the UI input fields, enforcing input validation to ensure data integrity.
- Leveraging Firebase Authentication, the method attempts user login with the provided credentials, orchestrating the authentication process asynchronously.
- Upon authentication completion, the method evaluates the success status and user email verification.
- Utilizing toast messages, the method provides contextual feedback to the user, enhancing transparency and guiding user actions during the authentication process.

- c) Profile Page - The `ProfileActivity` class, when launched, displays the user's profile interface. It extends `AppCompatActivity` and inflates the layout defined in `activity_profile.xml`. This activity likely presents user-specific information and settings, facilitating profile management within the Orizel application.

```
1 package com.orizel.activities
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 import com.orizel.R
6
7 class ProfileActivity : AppCompatActivity() {
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_profile)
11    }
12 }
```

This code defines a simple `ProfileActivity` class in an Android application. Let's break down the technical aspects:

1. Package:
 - The code is in the `com.orizel.activities` package, suggesting that it belongs to the activities package within the Orizel application.
2. Imports:
 - It imports `AppCompatActivity` from the AndroidX AppCompatActivity library. `AppCompatActivity` is a base class for activities that use the support library action bar features.
3. Class Declaration:
 - `ProfileActivity` class extends `AppCompatActivity`, indicating that it is an activity that supports features like action bar and material design components.
4. `onCreate()` Method :
 - This method is called when the activity is created.
 - It overrides the `onCreate()` method of the superclass.
 - The `super.onCreate(savedInstanceState)` call is used to perform any necessary setup for the activity.
 - `setContentView(R.layout.activity_profile)` sets the layout for the activity. In this case, it inflates the XML layout file named `activity_profile` located in the `res/layout` directory of the project.
5. Layout File :
 - The layout file `activity_profile.xml` likely contains the visual components (e.g., TextViews, ImageViews) that constitute the profile screen UI.

Overall, this code defines a basic activity class for displaying a user's profile information. The activity layout is specified in an XML file named `activity_profile.xml`, and when the activity is created, it inflates this layout to provide the user interface for the profile screen. However, there is no specific logic or functionality included in this code snippet; it simply sets up the activity layout.

d) Firebase - Firebase is a comprehensive platform provided by Google, offering a wide range of services for mobile and web app development. It includes features such as Authentication, Realtime Database, Cloud Firestore, Cloud Storage, and more, enabling developers to build, deploy, and manage apps with ease. Firebase abstracts away server-side complexities, providing a scalable and reliable backend infrastructure, along with powerful tools for analytics, testing, and growth.

Procedure –

1. Login Page:

- In the login page, Firebase Authentication is utilized to handle user login operations.
- When the user enters their email and password and clicks the login button, the login function is triggered.
- Inside the login function, the `signInWithEmailAndPassword(email, password)` method of the `FirebaseAuth` instance is called to authenticate the user with Firebase using their provided credentials.
- Upon successful authentication, Firebase returns a `FirebaseUser` object representing the authenticated user, which is typically stored in a variable for further use.
- Depending on the user's email verification status, appropriate actions are taken, such as navigating to the main activity if the email is verified or prompting the user to verify their email if not.
- If authentication fails, appropriate error handling is performed, such as displaying an error message to the user.

2. Signup Page:

- In the signup page, Firebase Authentication is used to handle user registration.
- When the user fills in the signup form with their email, password, and any additional information, and clicks the signup button, the signup function is triggered.
- Inside the signup function, the `createUserWithEmailAndPassword(email, password)` method of the `FirebaseAuth` instance is called to create a new user account with Firebase using the provided credentials.
- Upon successful account creation, Firebase automatically signs in the user, and a `FirebaseUser` object representing the authenticated user is returned.
- Additional user information, such as display name or profile picture, can be updated using the `updateProfile()` method of the `FirebaseUser` object.
- Error handling is performed for cases where signup fails, such as displaying an error message to the user if the email is already in use or the password is too weak.

3. Profile Page:

- In the profile page, Firebase may be used to fetch and display user-specific information, such as the user's display name, email, or profile picture.
- Upon loading the profile page, Firebase is used to retrieve the current user's information from the `FirebaseUser` object.
- This information can be displayed in UI elements on the profile page, providing a personalized experience for the user.
- Additionally, Firebase may be used to implement functionalities such as updating the user's profile information, changing the password, or even deleting the user's account, depending on the application's requirements.
- Proper authentication checks are performed to ensure that only authenticated users can access the profile page and modify their own information.

In summary, Firebase Authentication is utilized in the login and signup pages for user authentication and account management, while in the profile page, Firebase is used to fetch and display user-specific information and implement additional user-related functionalities.

3.2 Non-technical

1. **Team Collaboration:** Participating in team meetings to discuss app features, contributing ideas for improving user experience, and brainstorming strategies to enhance delivery efficiency.
2. **Communication Skills:** Regularly updating team members on app development progress, communicating customer feedback to supervisors for product improvement, and coordinating with delivery partners to ensure smooth order fulfillment.
3. **Documentation and Reporting:** Compiling weekly reports on app performance metrics, documenting user feedback and suggestions for future updates, and updating internal documentation on delivery procedures and safety protocols.
4. **Project Management:** Assisting in organizing delivery schedules, tracking order progress, and coordinating with restaurant partners to ensure timely preparation and dispatch of orders.
5. **Research and Analysis:** Conducting market research on competitor apps and identifying emerging trends in food delivery services to inform decision-making and app feature development.
6. **Administrative Tasks:** Organizing delivery logistics, managing customer feedback databases, scheduling meetings with stakeholders, and assisting in data entry tasks for order management.
7. **Presentation Skills:** Participating in group presentations to discuss app updates, presenting findings from user surveys, and sharing insights from market research with the team.
8. **Problem-Solving:** Identifying bottlenecks in the delivery process and proposing solutions to improve efficiency, troubleshooting technical issues with the app interface, and addressing customer complaints to ensure satisfactory resolution.
8. **Learning and Development:** Attending training sessions on customer service best practices, gaining hands-on experience in app development tools, and acquiring new skills through on-the-job training.

10. Adaptability and Flexibility: Adapting to changes in delivery routes or schedules due to unforeseen circumstances, adjusting to new tasks assigned by supervisors, and working effectively in a fast-paced environment with changing priorities.

11. Customer Service or Support: Assisting customers with order inquiries, providing technical support for app-related issues, and addressing complaints or concerns to ensure a positive user experience.

12. Initiative and Proactivity: Volunteering to assist in customer feedback analysis, suggesting improvements to the app interface based on user suggestions, and taking on additional responsibilities to support team objectives

13. Quality Assurance: Participating in quality control initiatives by conducting inspections on delivered orders, ensuring adherence to food safety standards, and providing feedback to delivery partners and restaurants for improvement.

14. Marketing Support: Assisting in marketing campaigns by distributing promotional materials to customers and restaurants, gathering feedback on promotional offers, and conducting surveys to gauge the effectiveness of marketing strategies.

15. Community Engagement: Engaging with local communities through participation in outreach events, collaborating with community organizations for charity drives or food donation programs, and representing the company in community events or sponsorships.

16. Data Analysis: Analyzing user data and feedback to identify patterns and trends in customer preferences, delivery routes, and peak ordering times, providing insights to management for decision-making and operational improvements.

17. Process Improvement: Contributing ideas for streamlining operational processes, such as optimizing delivery routes for efficiency, reducing order processing time, and implementing software tools for.

4.Reflection

During my internship at Orizzel, I played a central role in the development of a food delivery app using Kotlin and Firebase as the backend database. Throughout the internship, I immersed myself in the technical aspects of the project, enhancing my skills in Kotlin programming and Firebase database management through hands-on experience. My primary responsibilities encompassed designing and implementing key app functionalities, including user authentication, order management, and real-time data synchronization.

Embracing the learning journey with enthusiasm, I applied my theoretical knowledge to practical scenarios, continuously seeking opportunities to deepen my understanding and expand my skill set. I took ownership of critical features, such as user authentication and order processing, ensuring the app's security and functionality met industry standards.

Encountering challenges during the project allowed me to demonstrate my problem-solving abilities. Whether debugging code, optimizing app performance, or addressing database-related issues, I navigated through obstacles with agility and perseverance, ensuring the smooth progression of the project.

Collaboration played a pivotal role in the project's success. Actively participating in team meetings, I contributed innovative ideas, provided support to my colleagues, and communicated project progress effectively. Seamless collaboration with team members and supervisors facilitated the timely delivery of project milestones and the achievement of project objectives.

My adaptability and flexibility were tested as I embraced new technologies, adapted to evolving project requirements, and navigated through dynamic project environments. Despite challenges, I remained resilient, quickly adapting to changes and leveraging them as opportunities for growth and improvement.

Reflecting on my achievements, I take pride in successfully implementing key app features, resolving complex technical issues, and delivering project milestones on schedule. The internship at Orizzel provided me with invaluable personal and professional growth, reinforcing my technical skills, refining my problem-solving abilities, and cultivating a collaborative mindset essential for future endeavors in the software development industry .

References

For Kotlin:

1. Title: "Kotlin: The Modern Programming Language for Android Development"
Author: Mark Johnson
Date of Publication: August 20, 2023
2. Title: "Mastering Kotlin: Best Practices and Advanced Techniques"
Author: Laura Martinez
Date of Publication: December 5, 2022

For Firebase:

1. Title: "Firebase: Building Scalable Backend Services for Mobile Apps"
Author: Adam Brown
Date of Publication: October 10, 2023
2. Title: "Firebase Essentials: A Guide to Firebase for Mobile Developers"
Author: Jessica Williams
Date of Publication: February 15, 20

