Contents

Table of Content

GIT

Restaurant Table Booking App

Objectives

The main objective behind developing this kind of application for online restaurant table booking is to streamline the process of booking tables for customers and managing reservations for restaurants. The purpose of this system is to make reservations for restaurant tables more convenient and efficient for both customers.

Intended Users

- User

DFD

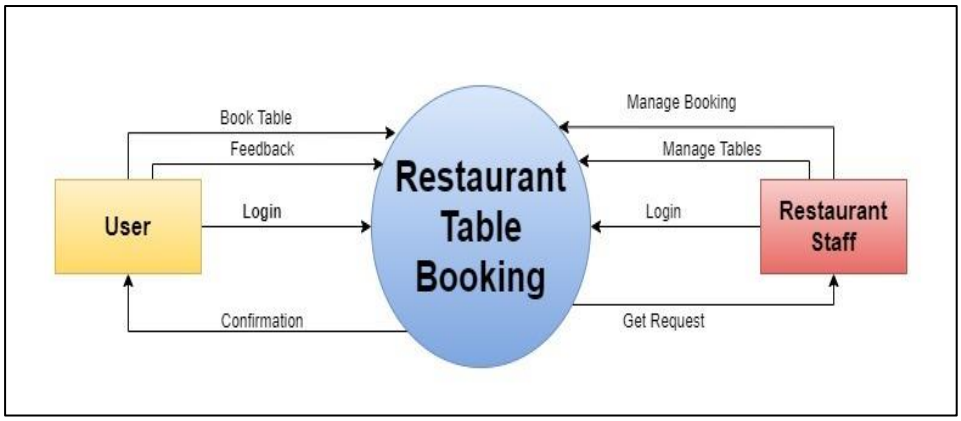


Figure 1: Data Flow Diagram

Sequence Diagram

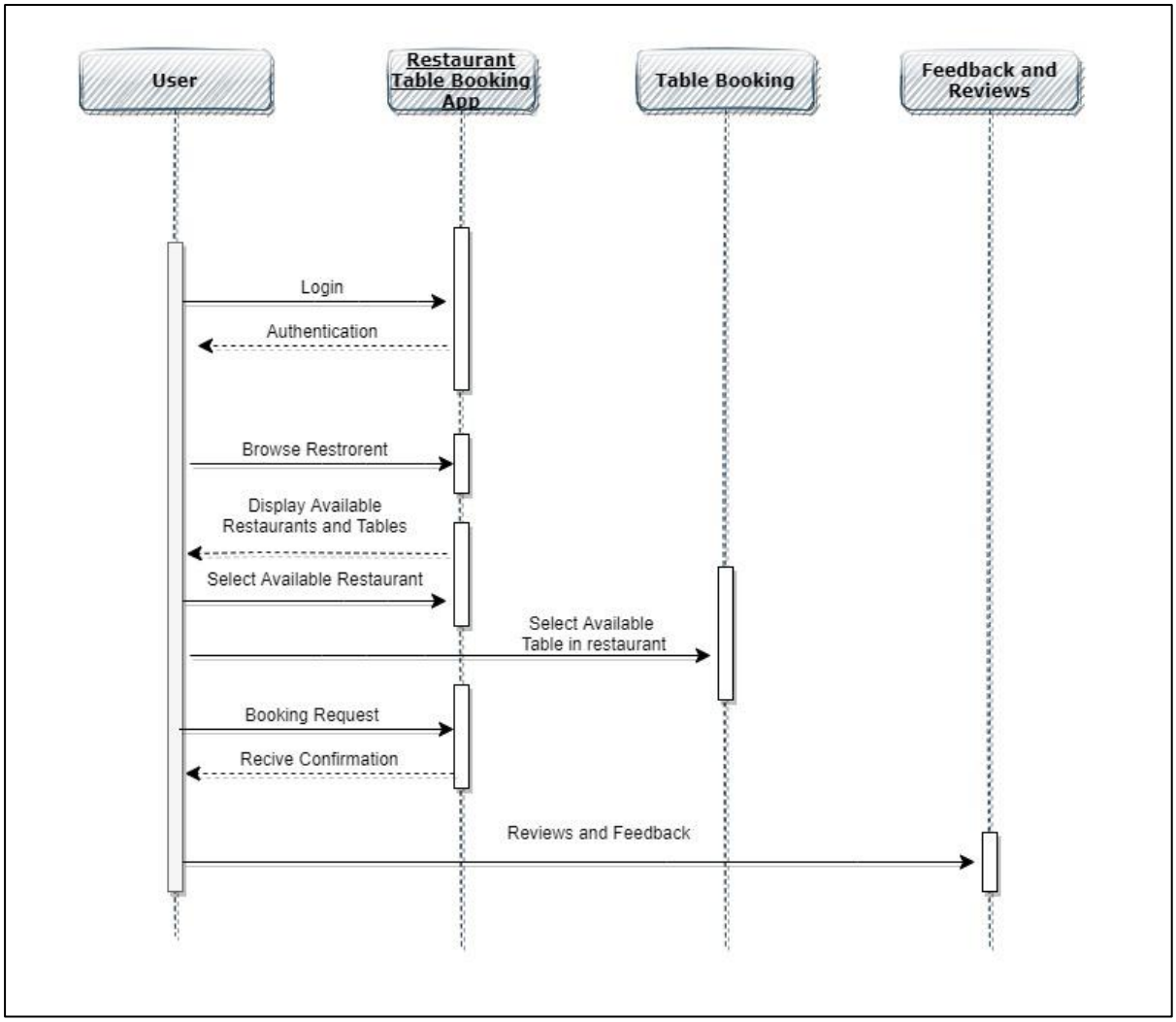


Figure 2: Sequence Diagram

Software Design Decisions

Tests were conducted on an emulator after the application was created with Android Studio. The application was developed using the Java programming language.

Justification of Software Design Decisions

Design & Layout:

To adapt to varying screen sizes, the majority of app activities employ a scroll view and linear style.

Navigator at the bottom and app's top are always the same.

Navigating bars slightly differently for instructor and student users.

User Experience:

Developed tutor users more highly than student users.

Features for the student session can be added following the section for the student user.

As more students asked for scheduling functionality, the introduction of the calendar was postponed.

Data Persistence:

Firebase Storage for profile picture storage.

Libraries Used and Code

```
package com.restauranttablebooking.routing

import androidx.compose.runtime.Composable
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import com.restauranttablebooking.ui.bookScreen.BookScreen
import com.restauranttablebooking.ui.detail.DetailScreen
import com.restauranttablebooking.ui.feedback.FeedbackScreen
import com.restauranttablebooking.ui.login.LoginScreen
import com.restauranttablebooking.ui.main.MainScreen
import com.restauranttablebooking.ui.register.RegisterScreen
import com.restauranttablebooking.ui.splash.SplashScreen
import com.restauranttablebooking.ui.thankScreen.ThankScreen

@Composable
fun Navigation() {
    val navController = rememberNavController()

    NavHost(
        navController = navController,
        startDestination = Screen.SplashScreen.route
    ) {
        composable(route = Screen.SplashScreen.route) {
            SplashScreen(navController = navController)
        }
        composable(route = Screen.LoginScreen.route) {
            LoginScreen(navController = navController)
        }
        composable(route = Screen.RegisterScreen.route) {
            RegisterScreen(navController = navController)
        }
        composable(route = Screen.MainScreen.route) {
            MainScreen(navController = navController)
        }
        composable(route = Screen.DetailScreen.route+ "/{name}"+"/{image}"+"/{address}"+"/{detail}") {
            val name = it.arguments?.getString("name")
            val image = it.arguments?.getString("image").toString().toInt()
            val address = it.arguments?.getString("address")
            val detail = it.arguments?.getString("detail")
            if (name != null) {
                if (image != null) {
                    if (address != null) {
                        if (detail != null) {
                            DetailScreen(navController = navController, name = name, image = image,detail = detail, address = address)
                        }
                    }
                }
            }
        }
```

Figure 3 Navigation routing Libraries Used and Code

```
package com.restauranttablebooking.routing

sealed class Screen(val route: String) {

    object SplashScreen: Screen("splash_screen")
    object LoginScreen: Screen("login_screen")
    object RegisterScreen: Screen("register_screen")
    object MainScreen: Screen("main_screen")
    object DetailScreen: Screen("detail_screen")
    object BookScreen: Screen("book_screen")
    object ThankScreen: Screen("thank_screen")
    object FeedbackScreen: Screen("feedback_screen")

}
```

Figure 4 Screen routing Libraries Used and Code

```
package com.restauranttablebooking.ui.main

import android.annotation.SuppressLint
import androidx.compose.foundation.*
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.rememberScaffoldState
import androidx.compose.material3.AlertDialog
import androidx.compose.material3.Card
import androidx.compose.material3.CardDefaults
import androidx.compose.material3.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.restauranttablebooking.R
import com.restauranttablebooking.routing.Screen
import com.restauranttablebooking.ui.drawer.DrawerBody
import com.restauranttablebooking.ui.drawer.DrawerHeader
import com.restauranttablebooking.ui.drawer.TopBar
import com.restauranttablebooking.ui.model.RestaurantModel
import com.restauranttablebooking.ui.restaurantPreference.RestaurantPreference
import com.restauranttablebooking.ui.theme.RestaurantTableBookingAppTheme
import com.restauranttablebooking.ui.theme.black
import com.restauranttablebooking.ui.theme.white
import com.restauranttablebooking.utils.RoundedButton
import kotlinx.coroutines.launch

@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter", "MutableCollectionMutableState")
@Composable
fun MainScreen(navController: NavController) {
    val context = LocalContext.current
    val preferenceManager = remember {
        RestaurantPreference(context)
    }
    val scaffoldState = rememberScaffoldState()
    val scope = rememberCoroutineScope()
    var isLogout by remember { mutableStateOf(false) }
    val scrollState = rememberScrollState()
```

Figure 5 MainScreen UI Libraries Used and Code

```
package com.restauranttablebooking.ui.feedback
import android.annotation.SuppressLint
import android.widget.Toast
import androidx.compose.foundation.*
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.selection.selectable
import androidx.compose.foundation.selection.selectableGroup
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Star
import androidx.compose.material.icons.rounded.ArrowBack
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.ExperimentalComposeUiApi
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.platform.LocalDensity
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.ImeAction
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.restauranttablebooking.R
import com.restauranttablebooking.ui.theme.RestaurantTableBookingAppTheme
import com.restauranttablebooking.ui.theme.black
import com.restauranttablebooking.ui.theme.white
import com.restauranttablebooking.utils.RoundedButton

@OptIn(ExperimentalMaterial3Api::class, ExperimentalComposeUiApi::class)
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@Composable
fun FeedbackScreen(navController: NavController) {
    val context = LocalContext.current
    var rate by remember { mutableStateOf(0F) }
    var isBooked by remember { mutableStateOf(false) }
    var review by remember { mutableStateOf("") }
    RestaurantTableBookingAppTheme {
        Scaffold {
            Column(
                modifier = Modifier
                    .fillMaxSize()
                    .background(color = black)
                    .padding(top = 40.dp)
            ) {
                SmallTopAppBar(
                    title = {
```

Figure 6 Feedbackcreen UI Libraries Used and Code

```
package com.restauranttablebooking.ui.login

import android.annotation.SuppressLint
import android.util.Log
import android.widget.Toast
import androidx.compose.foundation.*
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.ExperimentalComposeUiApi
import androidx.compose.ui.Modifier
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.compose.ui.window.Dialog
import androidx.compose.ui.window.DialogProperties
import androidx.navigation.NavController
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.firestore.ktx.firestore
import com.google.firebase.ktx.Firebase
import com.restauranttablebooking.R
import com.restauranttablebooking.routing.Screen
import com.restauranttablebooking.ui.restaurantPreference.RestaurantPreference
import com.restauranttablebooking.ui.theme.RestaurantTableBookingAppTheme
import com.restauranttablebooking.ui.theme.black
import com.restauranttablebooking.ui.theme.white
import com.restauranttablebooking.utils.RestaurantTableBookingBorderFeild
import com.restauranttablebooking.utils.RoundedButton
import com.restauranttablebooking.utils.isValidEmail

@OptIn(ExperimentalMaterial3Api::class, ExperimentalComposeUiApi::class)
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@Composable
fun LoginScreen(navController: NavController) {
    val context = LocalContext.current
    val preference = remember {
        RestaurantPreference(context)
    }
```

Figure 7 LoginScreen UI Libraries Used and Code

```
package com.restauranttablebooking.ui.restaurantPreference

import android.content.Context
import android.content.SharedPreferences

class RestaurantPreference(context: Context) {
    private val sharedPreferences: SharedPreferences =
        context.getSharedPreferences("grocery_database", Context.MODE_PRIVATE)

    fun saveData(key: String, value: Boolean) {
        val editor = sharedPreferences.edit()
        editor.putBoolean(key, value)
        editor.apply()
    }

    fun getData(key: String, defaultValue: Boolean = false): Boolean {
        return sharedPreferences.getBoolean(key, defaultValue) ?: defaultValue
    }
}
```

Figure 8 RestaurantPreference UI Libraries Used and Code

```
package com.restauranttablebooking.ui.thankScreen

import android.annotation.SuppressLint
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.rounded.ArrowBack
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.restauranttablebooking.R
import com.restauranttablebooking.routing.Screen
import com.restauranttablebooking.ui.restaurantPreference.RestaurantPreference
import com.restauranttablebooking.ui.theme.RestaurantTableBookingAppTheme
import com.restauranttablebooking.ui.theme.black
import com.restauranttablebooking.ui.theme.white
import com.restauranttablebooking.utils.RoundedButton
import kotlinx.coroutines.delay
import kotlin.time.Duration.Companion.seconds

@OptIn(ExperimentalMaterial3Api::class)
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@Composable
fun ThankScreen(navController: NavController) {

    RestaurantTableBookingAppTheme {
        Scaffold {
            Column(
                modifier = Modifier
                    .background(color = black)
                    .padding(top = 40.dp)
            ) {
                SmallTopAppBar(
                    title = {
                        Text(
                            text = "Thanks Screen", color = Color.White,
                            modifier = Modifier
                                .fillMaxWidth()
                                .padding(10.dp),
```

Figure 9 ThankScreen UI Libraries Used and Code

```
package com.restauranttablebooking.utils

import java.util.regex.Pattern

fun isValidEmail(s: String): Boolean {
    val emailPattern = ("^(([\\w-]+\\.)+[\\w-]+|([a-zA-Z]{1}|[\\w-]{2,}))@"
            + "((([0-1]?[0-9]{1,2}|25[0-5]|2[0-4][0-9])\\.([0-1]?"
            + "[0-9]{1,2}|25[0-5]|2[0-4][0-9])\\."
            + "([0-1]?[0-9]{1,2}|25[0-5]|2[0-4][0-9])\\.([0-1]?"
            + "[0-9]{1,2}|25[0-5]|2[0-4][0-9])){1}|"
            + "([a-zA-Z]+[\\w-]+\\.)+[a-zA-Z]{2,4})$")
    val pattern = Pattern.compile(emailPattern)
    val matcher = pattern.matcher(s)

    return !matcher.matches()
}
```

Figure 10 Extension utils Libraries Used and Code

```
package com.restauranttablebooking.utils

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Brush
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.restauranttablebooking.ui.theme.black

@Composable
fun RoundedButton(text: String, isEnabled : Boolean = true, onClick: () -> Unit, textColor : Color = Color.White) {
    GradientButton(
        onClick = onClick,
        modifier = Modifier.padding(vertical = 5.dp).fillMaxWidth(),
        textColor = textColor,
        isEnabled = isEnabled,
        gradient = Brush.horizontalGradient(listOf(black, black)),
        text = text
    )
}

@Composable
fun GradientButton(
    text: String,
    gradient : Brush,
    modifier: Modifier = Modifier,
    onClick: () -> Unit = { },
    textColor: Color,
    isEnabled: Boolean
) {
    Button(
        modifier = modifier,
        colors = ButtonDefaults.buttonColors(containerColor = Color.Transparent, contentColor = textColor),
        contentPadding = PaddingValues(),
        shape = RoundedCornerShape(30.dp),
        enabled = isEnabled,
        onClick = { onClick() },
```

Figure 11 RoundedButton utils Libraries Used and Code

```
package com.restauranttablebooking.utils

import androidx.compose.foundation.clickable
import androidx.compose.foundation.focusable
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardActions
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.ExperimentalComposeUiApi
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalSoftwareKeyboardController
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.KeyboardCapitalization
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.unit.Dp
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.restauranttablebooking.ui.theme.black

@OptIn(ExperimentalMaterial3Api::class, ExperimentalComposeUiApi::class)
@Composable
fun RestaurantTableBookingBorderFeild(
    value: String,
    placeholder: String,
    onValueChange: (String) -> Unit,
    onClick: () -> Unit = {},
    isEnabled: Boolean = true,
    readOnly: Boolean = false,
    keyboardType: KeyboardType = KeyboardType.Text,
    keyboardCapitalization: KeyboardCapitalization = KeyboardCapitalization.Sentences,
    visualTransformation: VisualTransformation = VisualTransformation.None,
    leadingIcon: @Composable (() -> Unit)? = null,
    trailingIcon: @Composable (() -> Unit)? = null,
    horizontalPadding : Dp = 0.dp,
    verticalPadding : Dp = 3.dp,
    colors : TextFieldColors? = null,
    label : @Composable (() -> Unit)? = null,
) {
    val keyboardController = LocalSoftwareKeyboardController.current

    OutlinedTextField(
        value = value,
        shape = RoundedCornerShape(10.dp),
        keyboardOptions = KeyboardOptions(keyboardType = keyboardType, capitalization = keyboardCapitalization),
        keyboardActions = KeyboardActions(
            onDone = {keyboardController?.hide()}),
        visualTransformation = visualTransformation,
        enabled = isEnabled,
```

Figure 12 RestaurantTableBookingBorderFeild utils Libraries Used and Code

```
package com.restauranttablebooking

import android.os.Bundle
import android.view.WindowManager
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.restauranttablebooking.ui.theme.RestaurantTableBookingAppTheme
import com.restauranttablebooking.routing.Navigation

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        window.setFlags(
            WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
            WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS
        )
        setContent {
            Navigation()
        }
    }
}

@Composable
fun Greeting(name: String) {
    Text(text = "Hello $name!")
}

@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    RestaurantTableBookingAppTheme {
        Greeting("Android")
    }
}
```

Figure 13 MainActivity Libraries Used and Code

Social, Legal, Ethical and Security Issue

Figure 14: Social, legal, ethical and security issues

Social Issues

1. Equitable and Fair Access: -

The Restaurant Table Booking App should guarantee its user to fair and Equitable access to table reservations system via application. The discrimination based on user geography, socioeconomic level and other characteristics should not be happening via this application.

2. Anxiety for Table Selection: -

The main and fundamental goal of this Restaurant Table Booking app should be to decrease the user pressure to reserve the "perfect" table because it is always the anxiety problem to the user to book suitable table in the restaurant.

3. Participation of the Community: -

For the betterment and to respect the user opinion, the Restaurant Table Booking App need to be tried to provide rating, review, and feedback system to all its users.

Legal Issues

4. Compliance with Reservation Laws: -

The Restaurant Table Booking App must need to follow the local authorities and restaurant laws about the restaurant bookings about the no-show policies, reservation cancellation procedures, and accessibility for people with disabilities. . The laws that define the nourishment of this schema are-

Consumer Contracts Regulations (2014)

The Cancellation of Contracts Made by Distance Selling Regulations (2000)

5. External APIs: -

The Restaurant Table Booking App need to be stand clear for the policies that uses any third party the external APIs because in future, they can create a cause for the copyright issue and defamation according to the **Consumer Rights Act 2015.**

6. Intellectual Property Right: -

The Restaurant Table Booking Application must need to be mentioned that this application does not infringe on any Intellectual Property Right act, violation of the trademarks and copyrights laws of restaurants and other entities.

Ethical Issues

7. Transparency: -

The Restaurant Table Booking App should furnish users with clear and understandable information regarding the restaurant policies, table availability in restaurant, table size, booking priority and pricing so that customers can able to make well-informed choices about their reservation.

8. Respect for Restaurant Partners: -

The Restaurant Table Booking App should need to try to respect the interests and concerns of the restaurant by giving the restaurant partners authority over their booking policies, availability, and data sharing procedures within the app.

9. Accessibility for Walk-Ins: -

The Restaurant Table Booking App should try to deliver the instruction the user to strike a balance between accepting online reservations and serving to individuals who come in.

Security Issues

10. Reservation Security: -

In order to prevent the restaurant partners from making losses and to preserve the integrity of the booking system via application, the Restaurant Table Booking App

should try to enable strong procedures to avoid fraudulent and repeatable booking at the same time for the same user.

11. Denial-of-Service Attacks: -

The Restaurant Table Booking Application need to be resistant to denial-of-service attacks. It is important for such booking application because the DoS attack may have the potential to interfere with booking operations.

12. Secure and Valid Authentication: -

The restaurant table booking application must try to use strong and robust authentication processes that can verify the user identities and should be able to prevent unauthorized access to the user accounts.

Development Plan:

We apply Agile Methodology to build the app. Making any changes and adding features or data becomes easy in using agile method.

Sprint Goal:

The below screenshot contain the sprint goal.

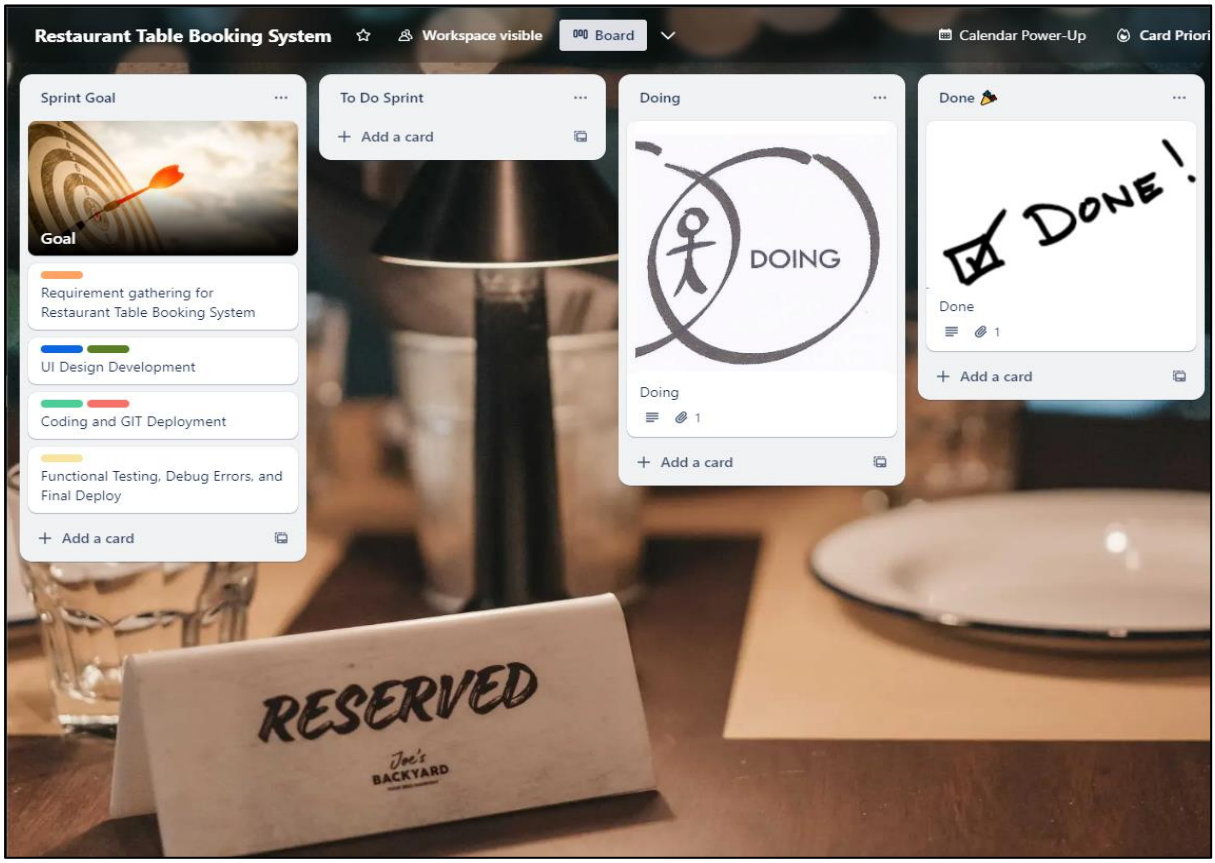


Figure 15: Sprint Goal

Phase-1:

The tasks are drawn for sprints. For timely completion, we make a deadline on each sprint. The sprint tasks are given in the below screenshots.

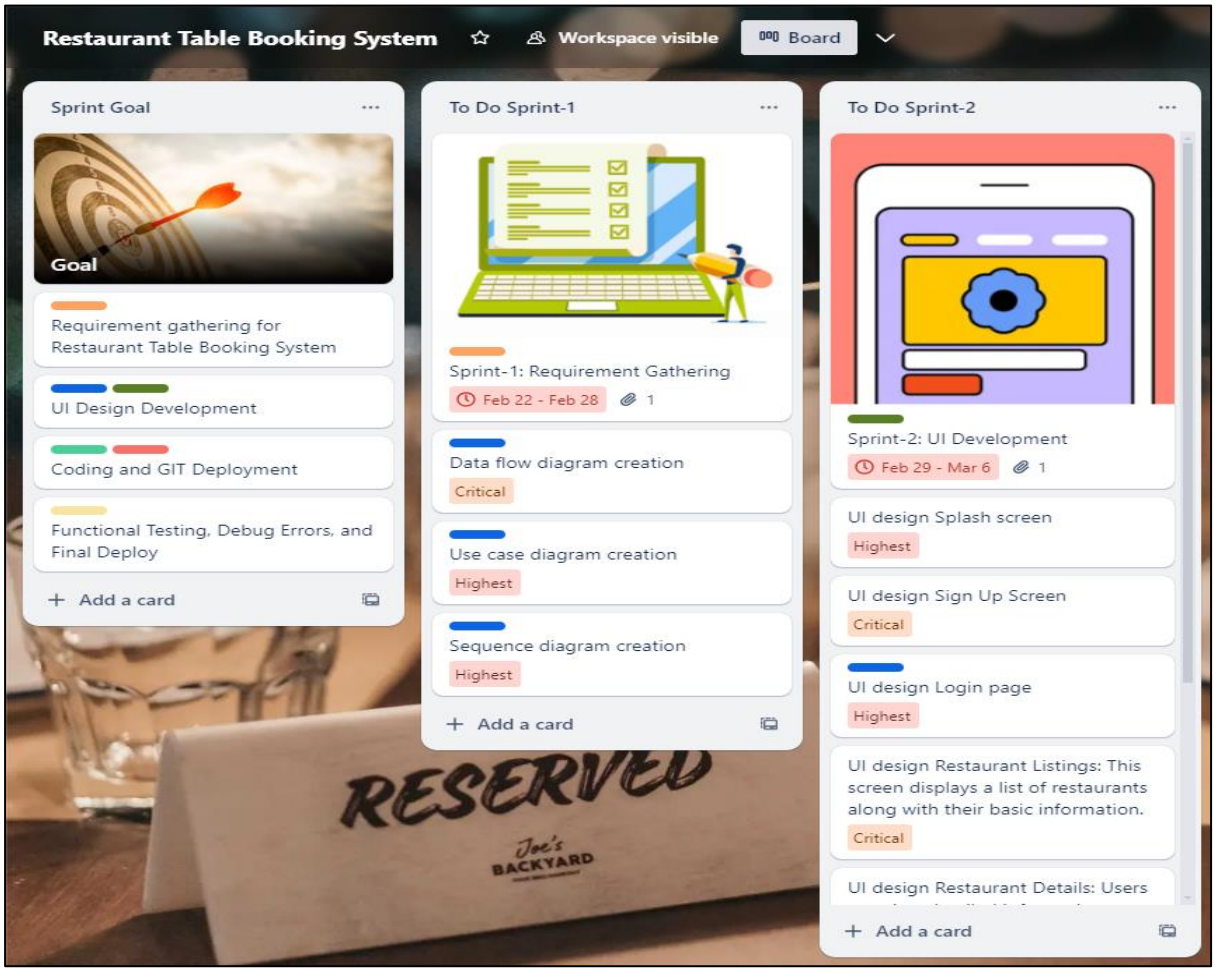


Figure 16: Phase I (i)

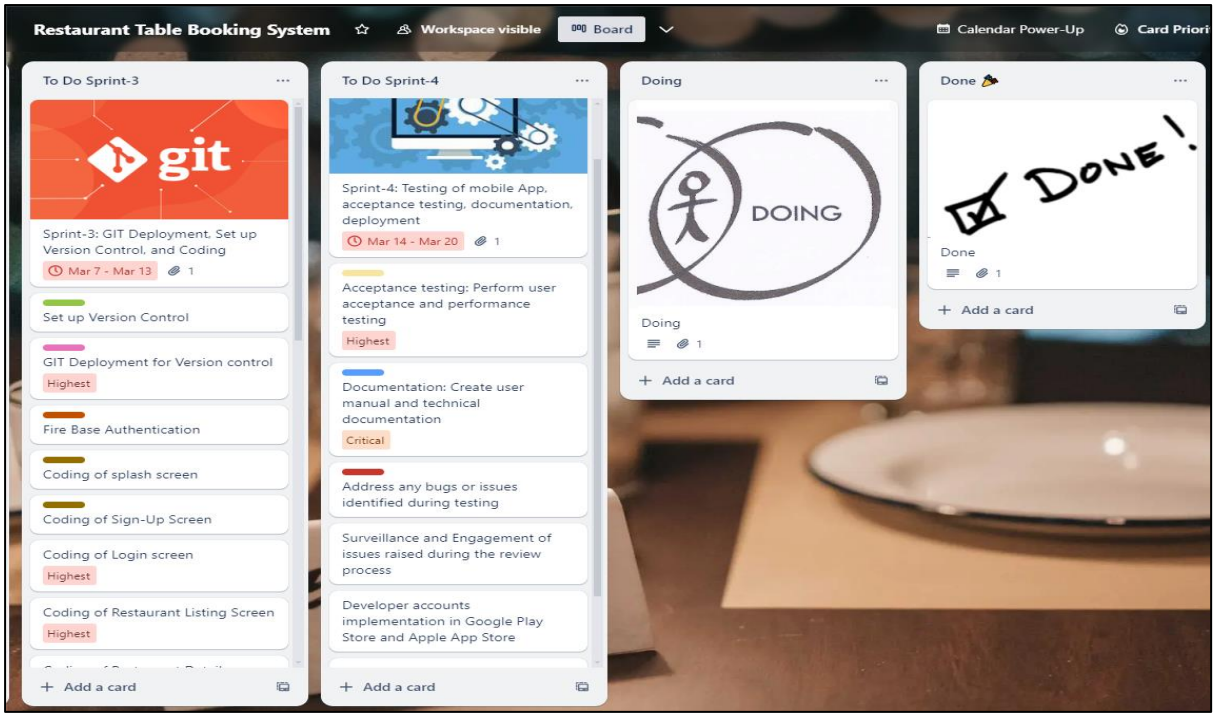


Figure 17: Phase I (ii)

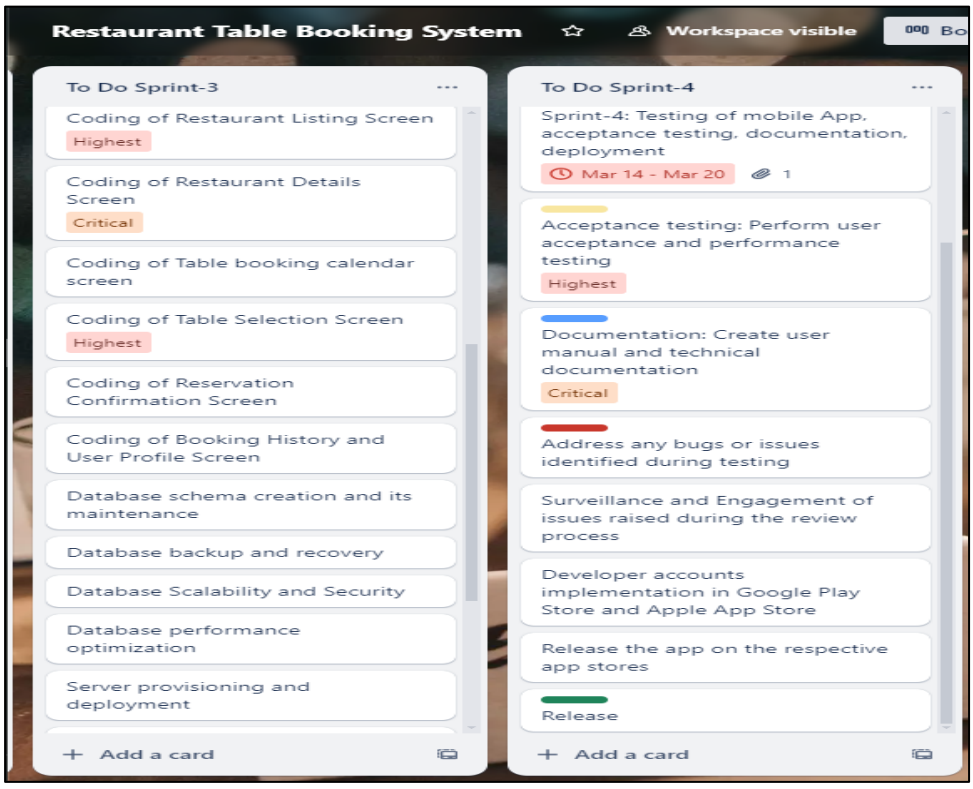


Figure 18: Phase I (iii)

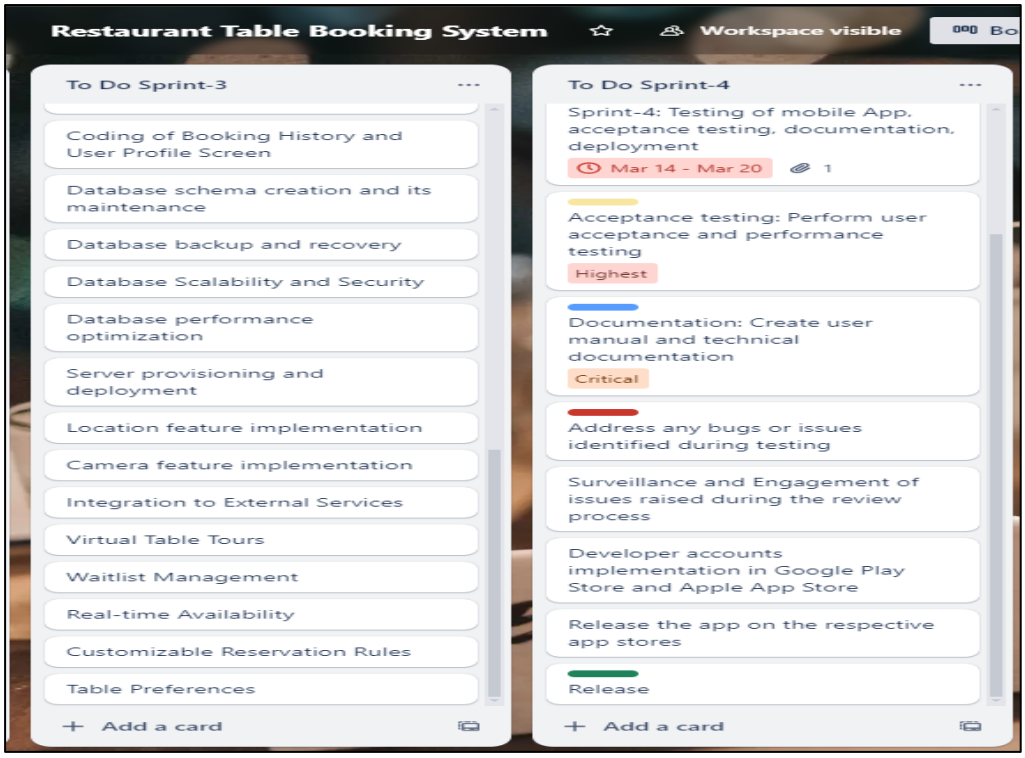


Figure 19: Phase I (iv)

Phase-2: This phase contains the tasks which are ongoing. Among these tasks, some are impossible to achieve. Therefore, they have been moved to Abandoned Tasks List.

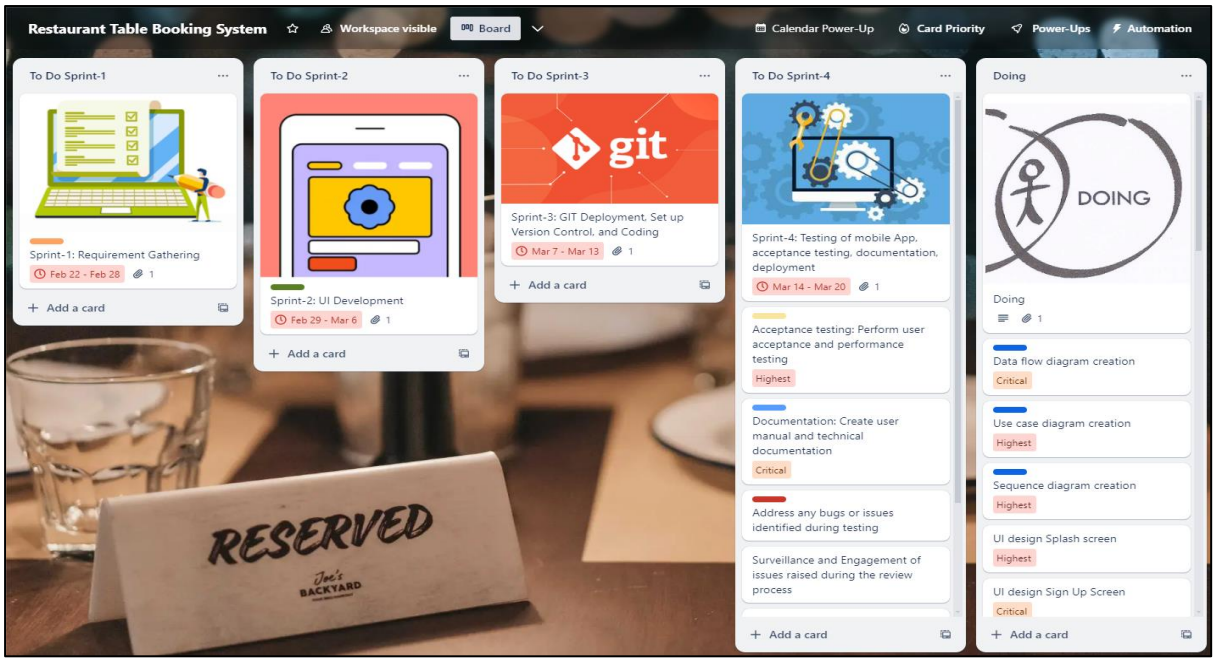


Figure 20: Phase II

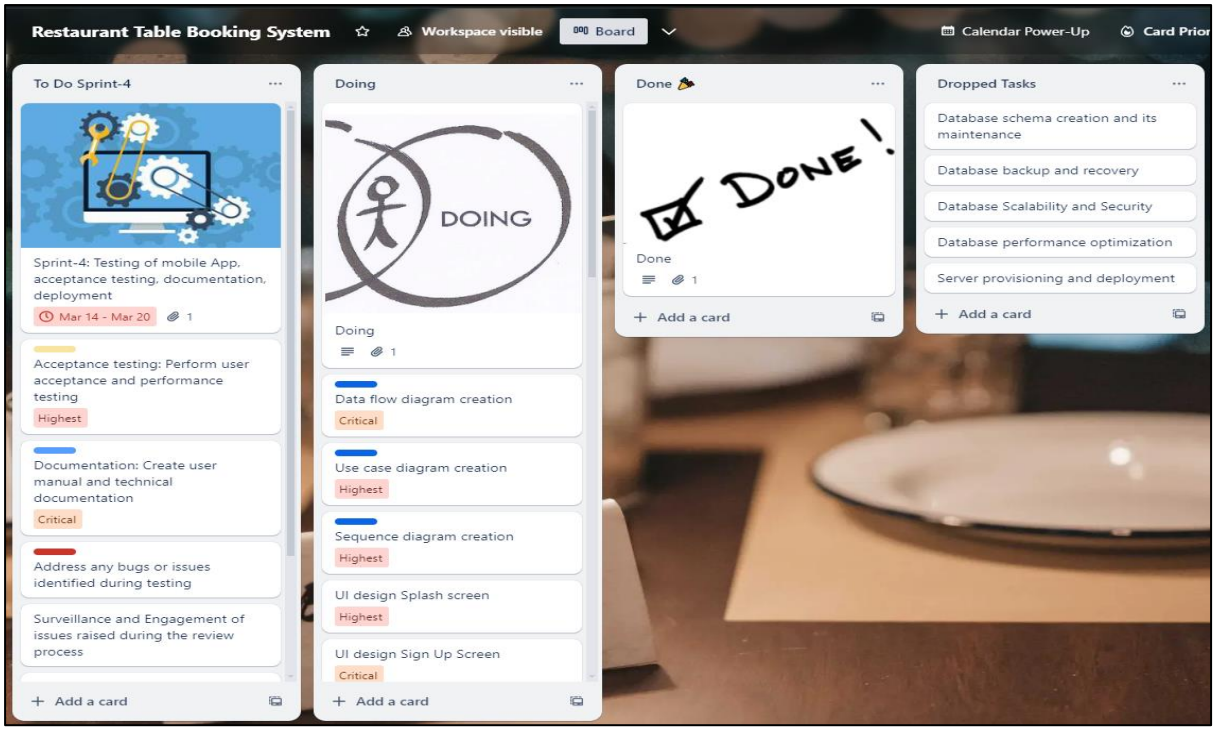


Figure 21: Phase II (ii)

Phase-3: We move all completed tasks to done list. The app testing is started then. However, certain tasks are not completed, hence, they have been put in Not Reachable Tasks List.

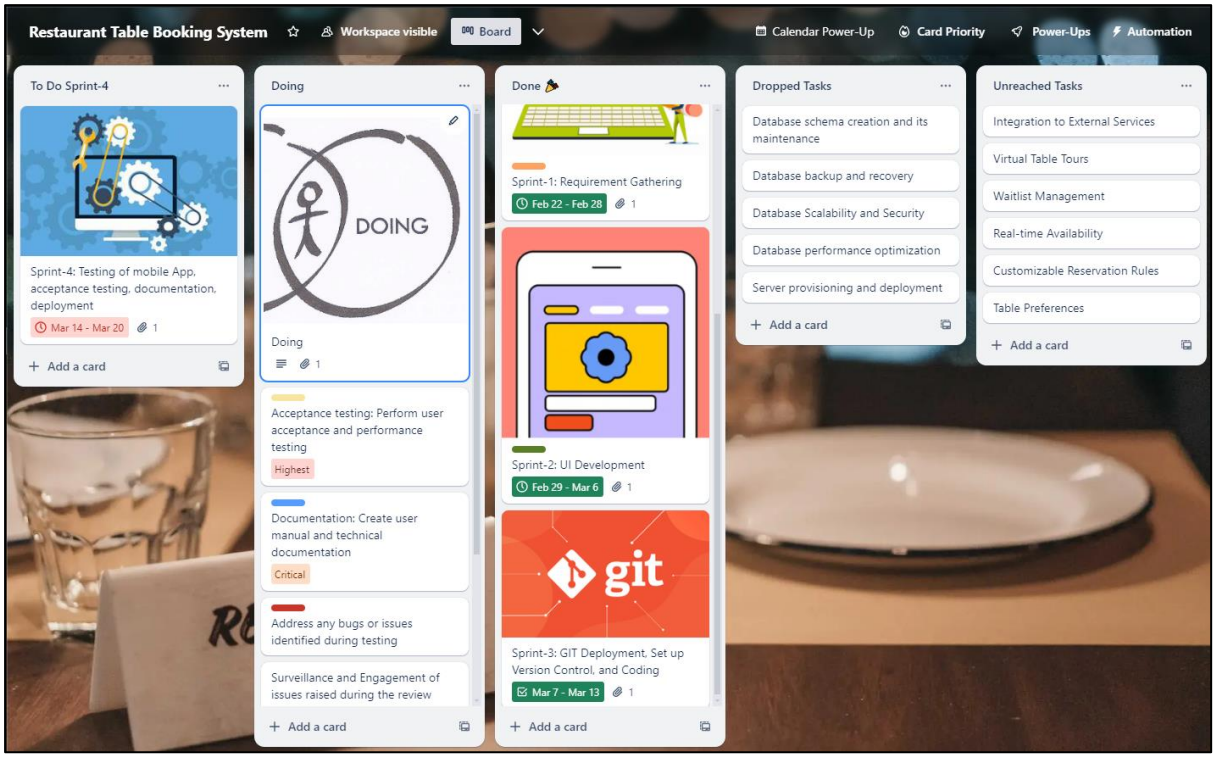


Figure 22: Phase III

Phase-4: In this phase, testing is completed. Now, app is released.

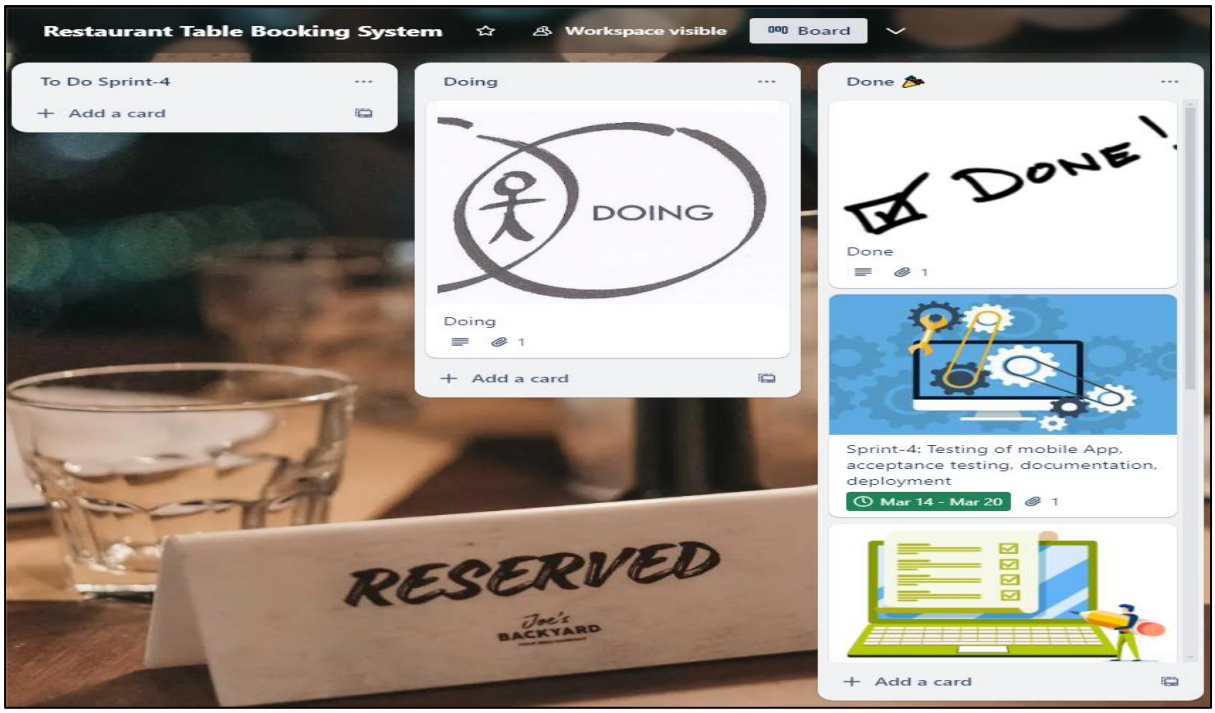


Figure 23: Phase IV (i)

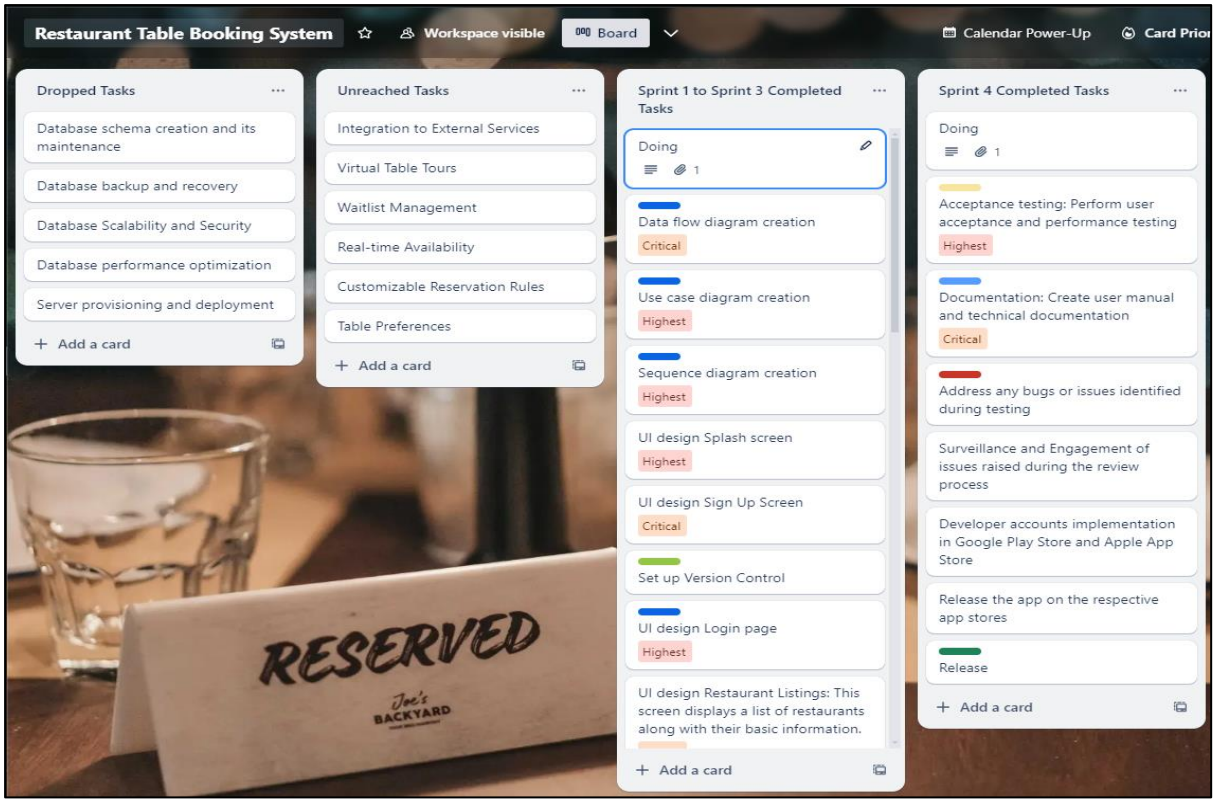


Figure 24: Phase IV (ii)

Burn down Charts: The burn down charts for each sprint are given by:

Burn down Chart Sprint 1:

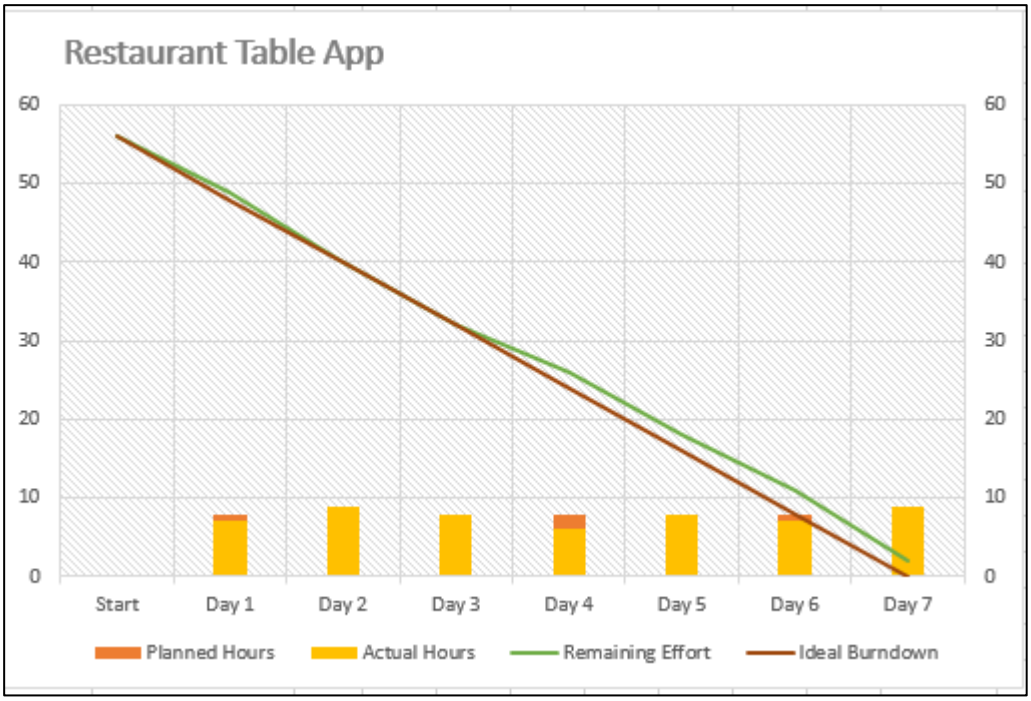


Figure 25: Burn Down Chart Sprint I

Burn down Chart Sprint 2:

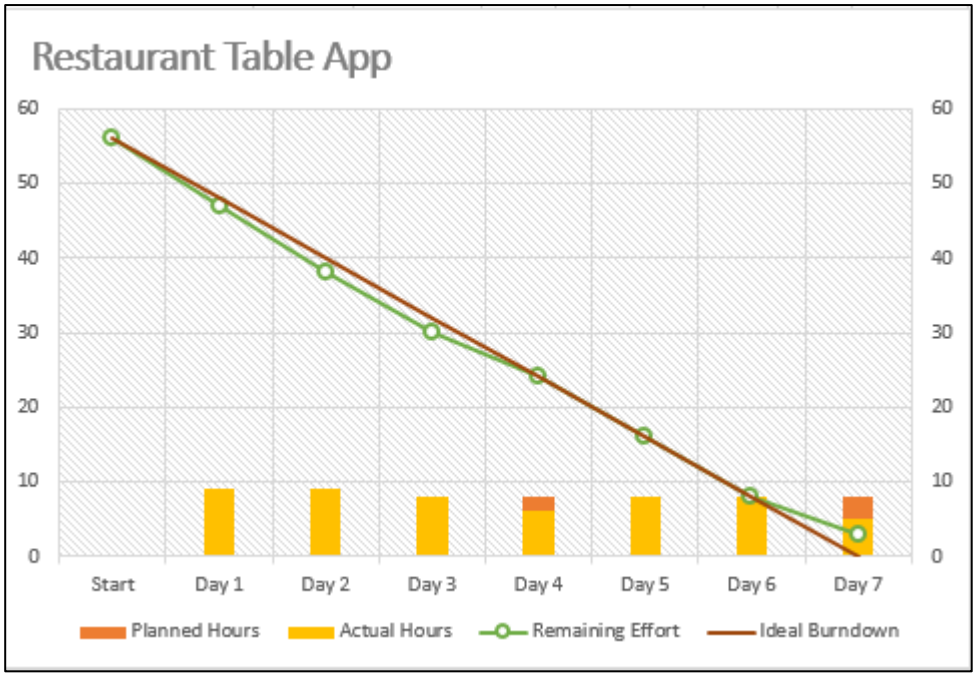


Figure 26: Burn Down Chart Sprint II

Burn down Chart Sprint 3:

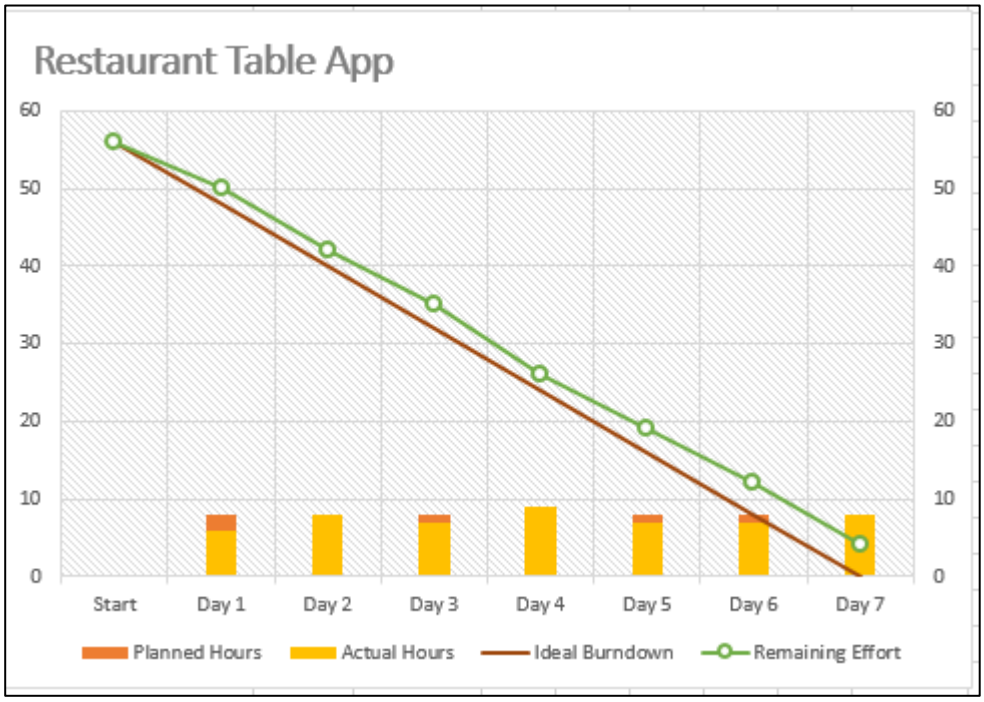


Figure 27: Burn Down Chart Sprint III

Burn down Chart Sprint 4:

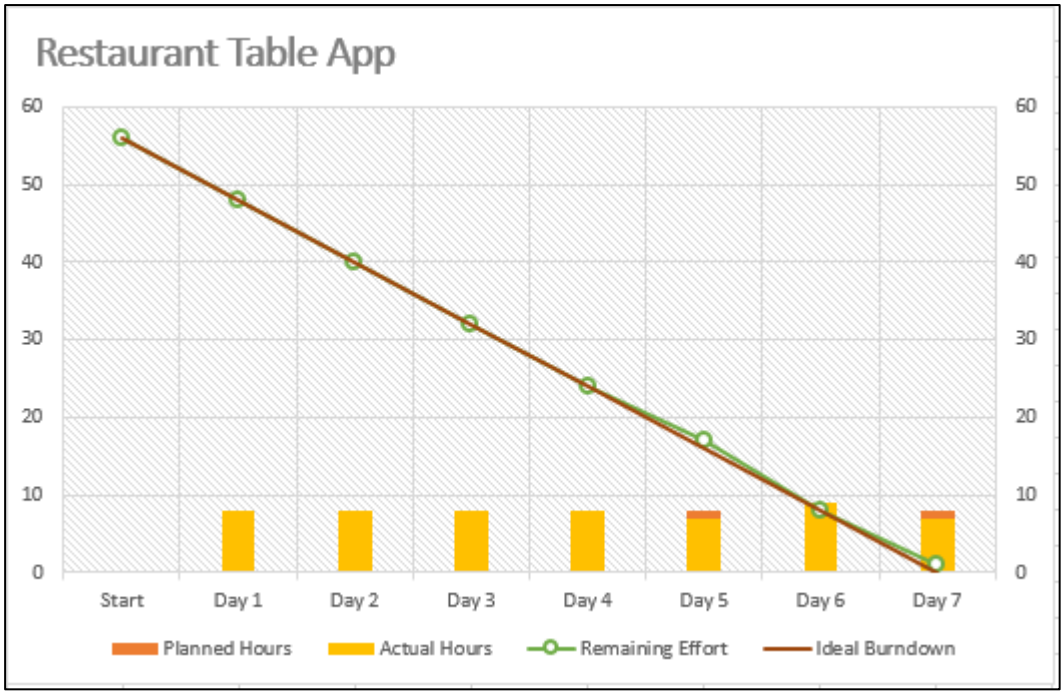


Figure 28: Burn Down Chart Sprint IV

Firebase DB at backend and regarding API authentication and searching

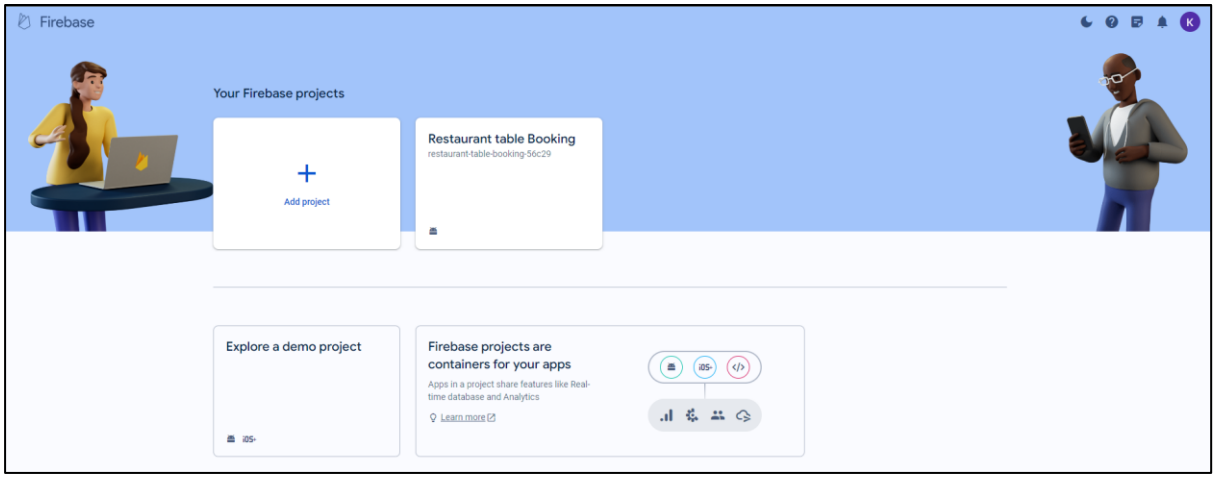


Figure 29 Firebase DB

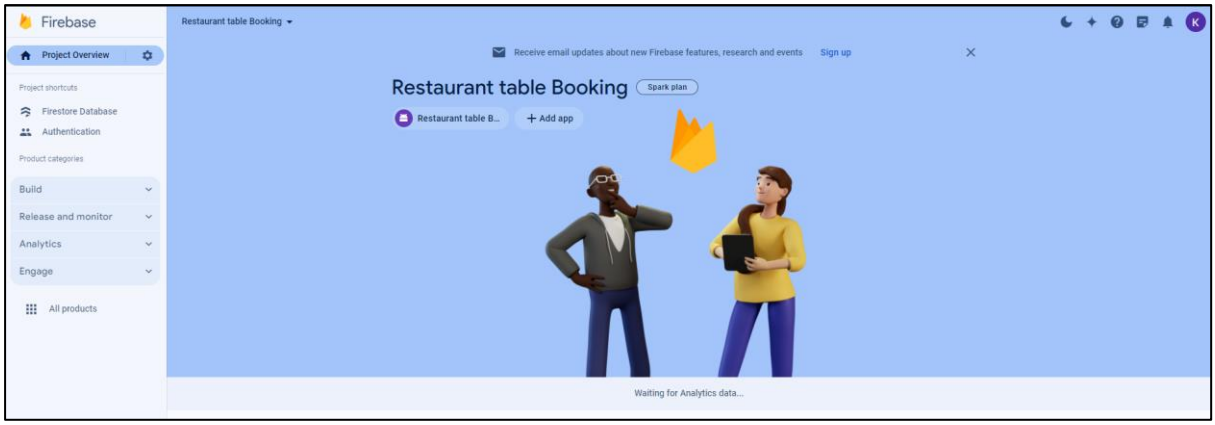


Figure 30 Firebase Restaurant table booking app overview

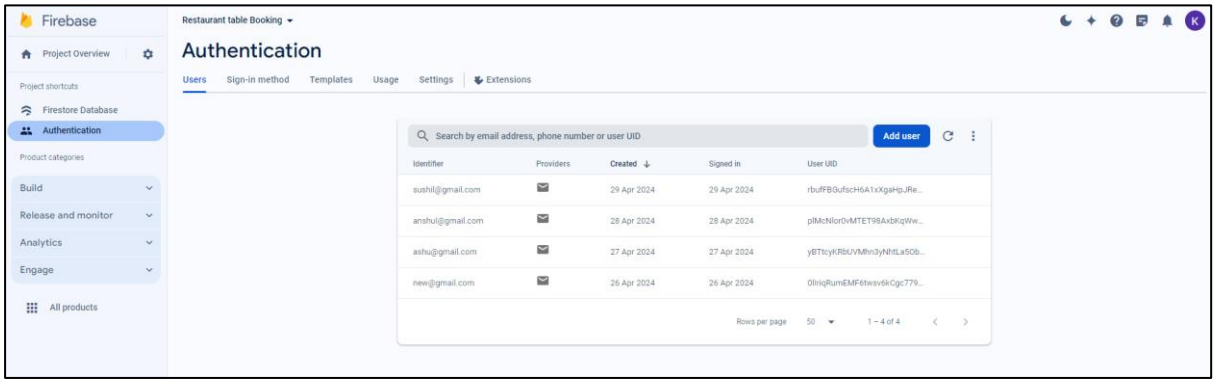


Figure 31 Firebase authentication

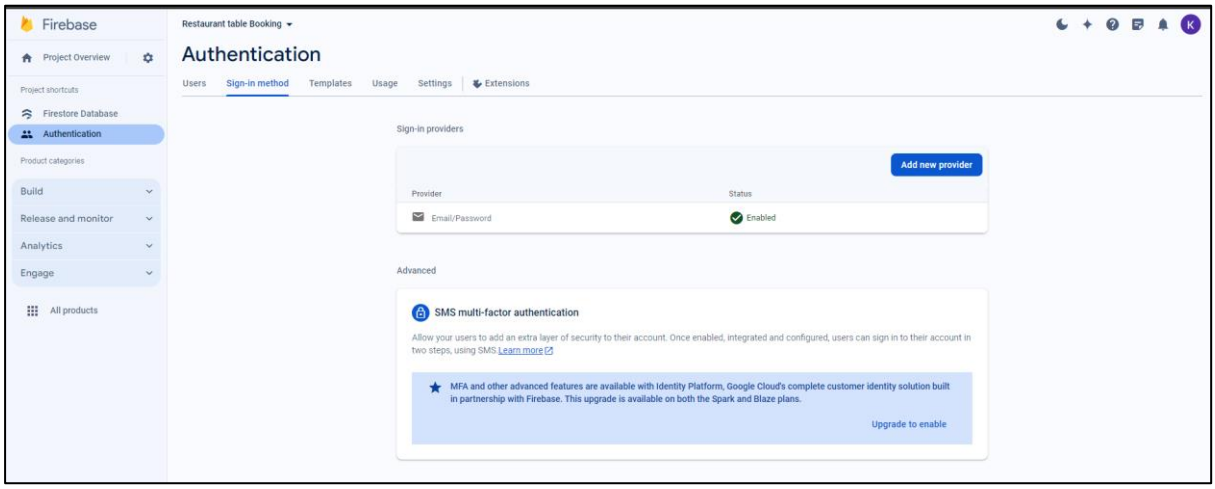


Figure 32 Sign in methods in firebase part 1

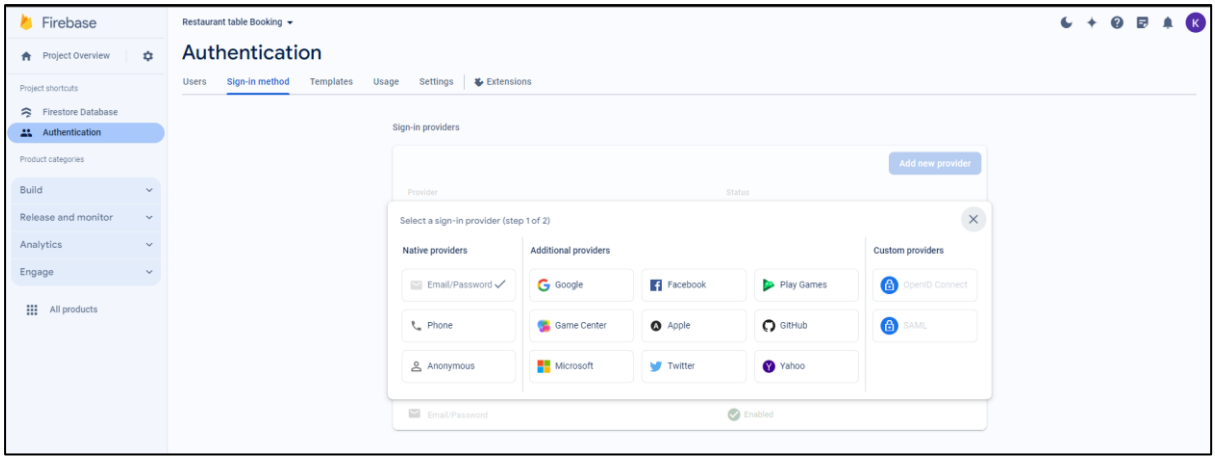


Figure 33 Gmail log in method in firebase part 2

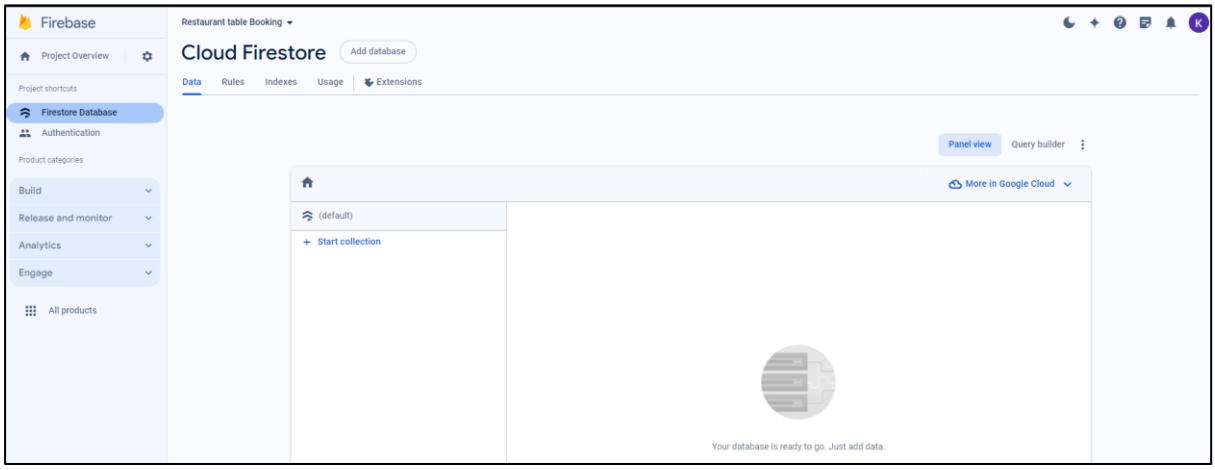


Figure 34 Cloud firestore

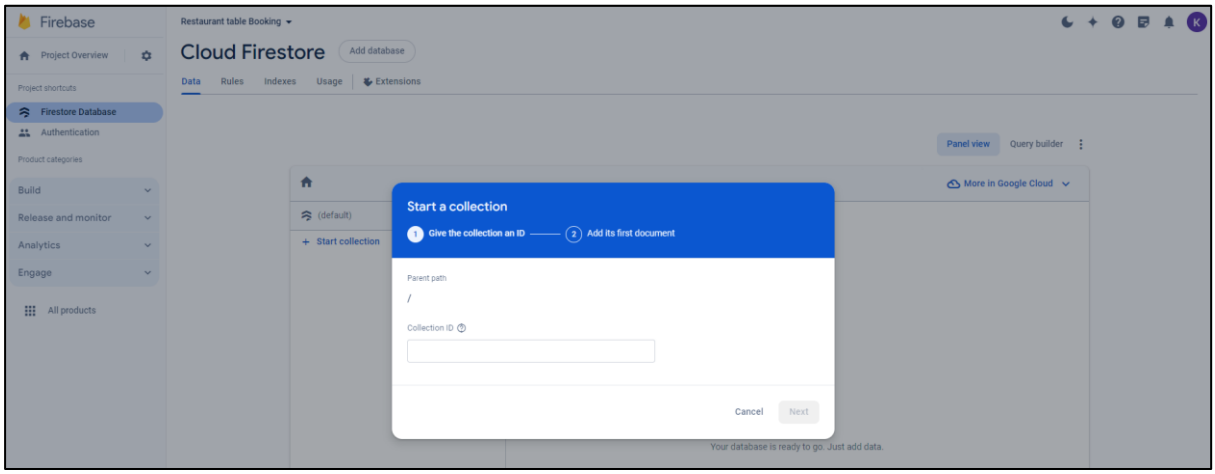


Figure 35 Collection starting in DB

Restaurant Table Booking System (UI)

1. Welcome Page –

Figure 36:Welcome Page

2. Registration Page –

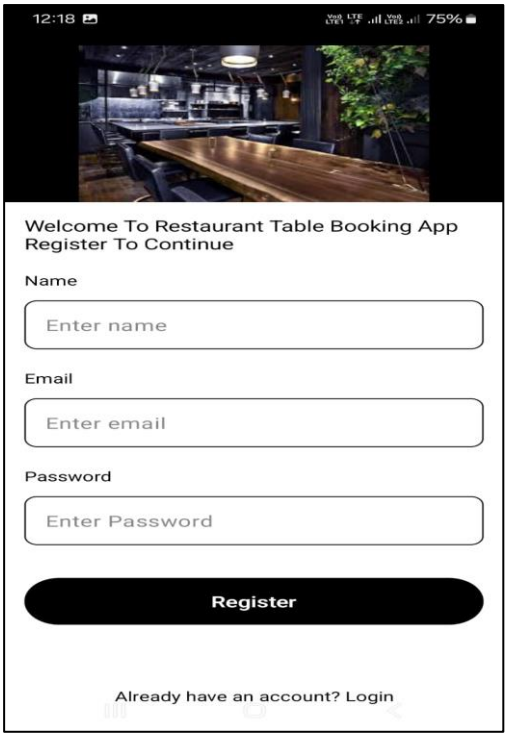


Figure 37: Registration Page

3. Login Page –

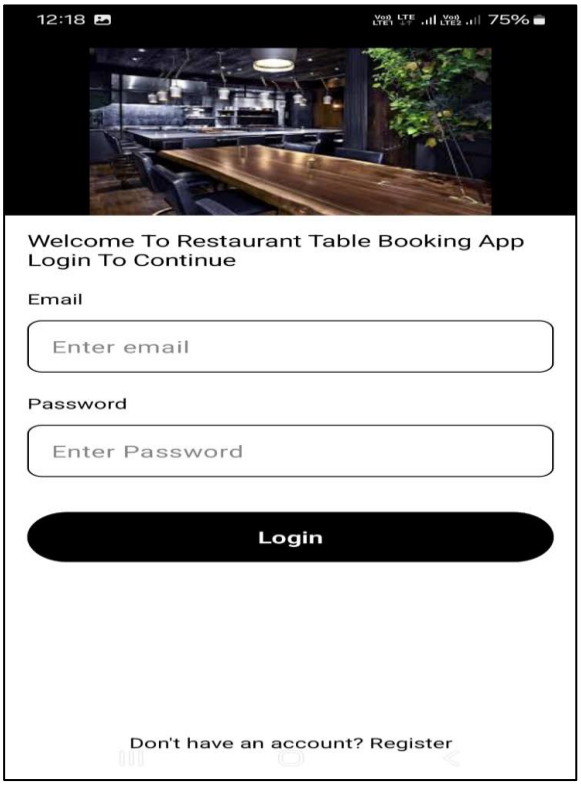


Figure 38: Login Page

4. Home Page –

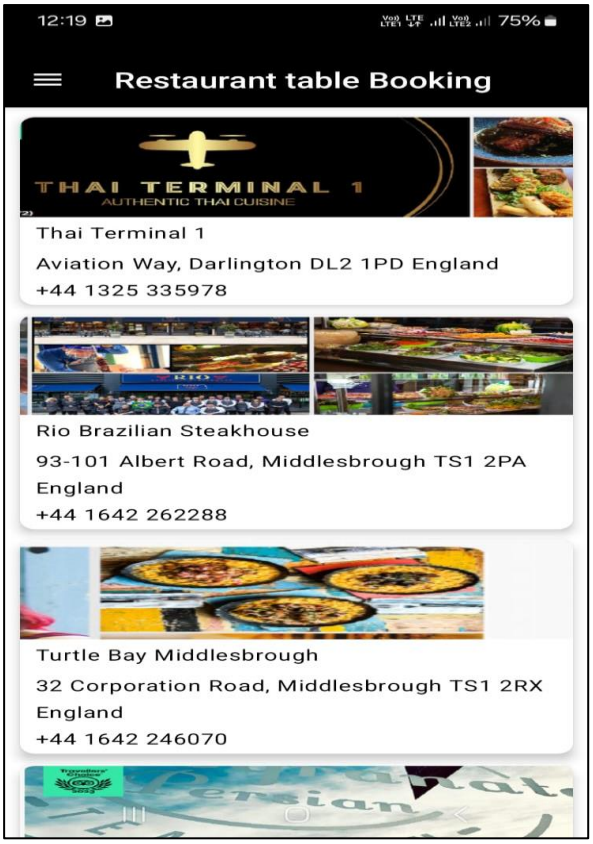
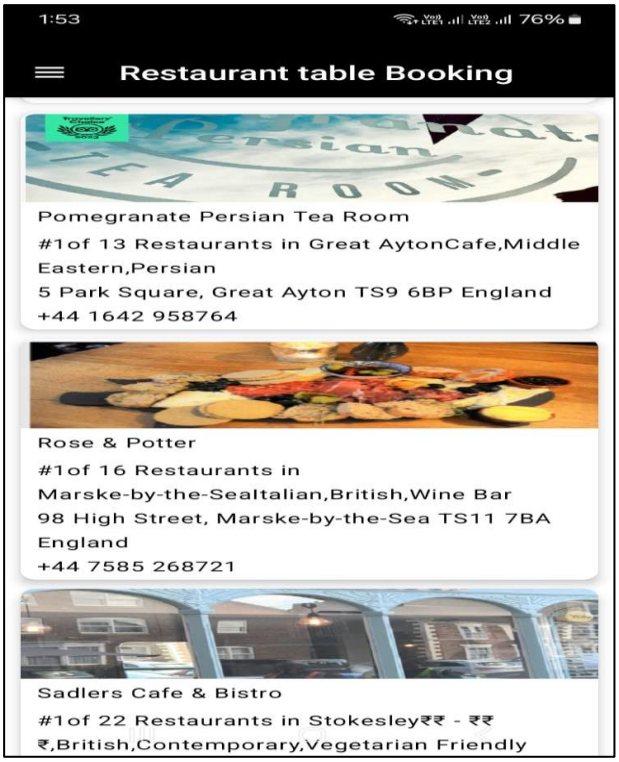


Figure 39: Home Page



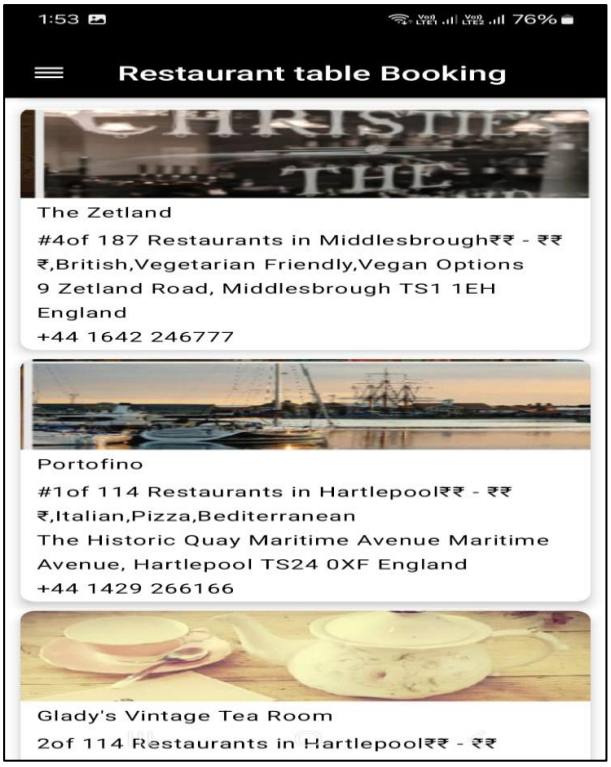


Figure 40: Home Page II

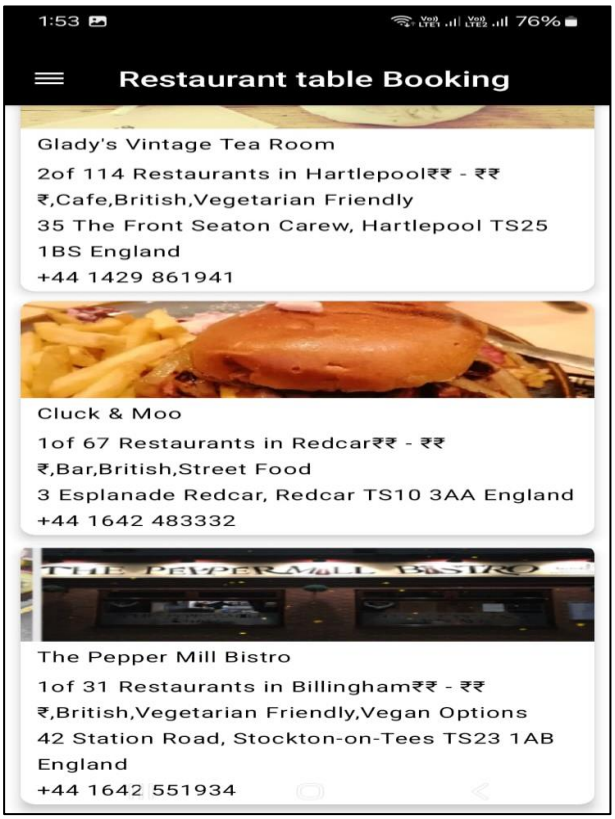


Figure 41: Home Page III

5. Details Page –

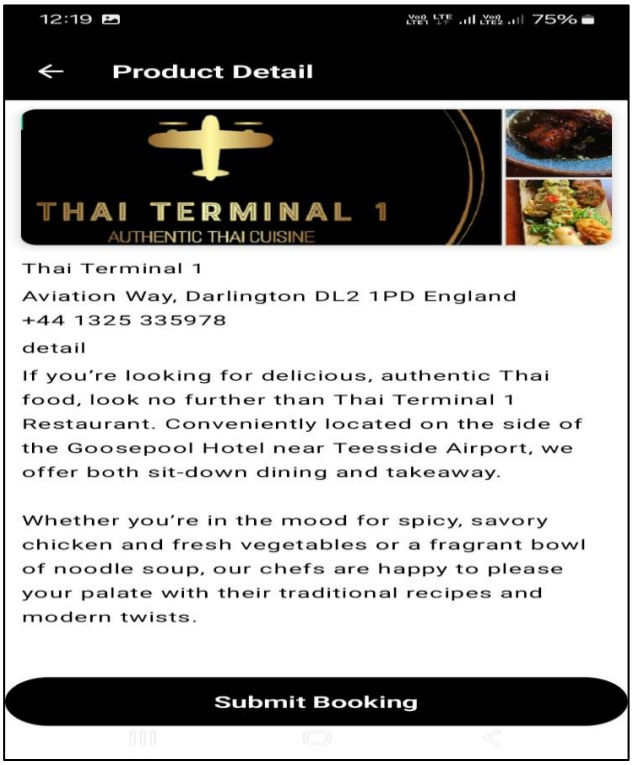


Figure 42: Details Page

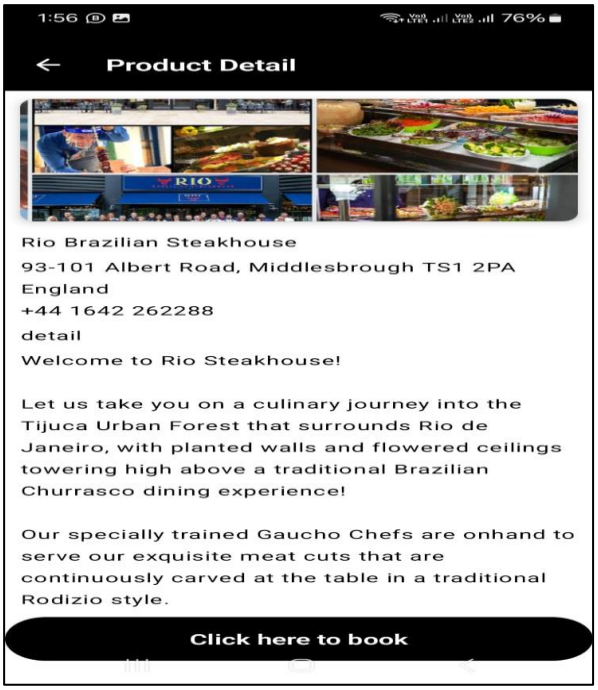


Figure 43: Details Page II

Figure 44: Detail Page III

Figure 45: Details Page IV

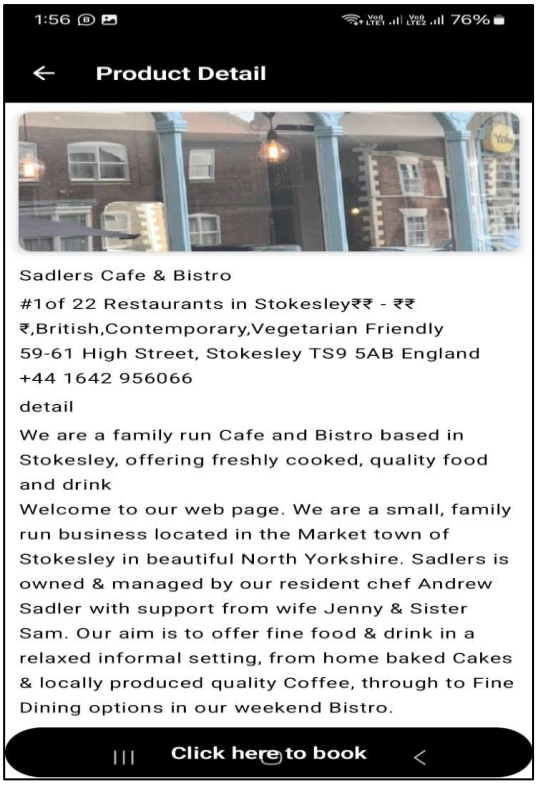


Figure 46: Details Page V

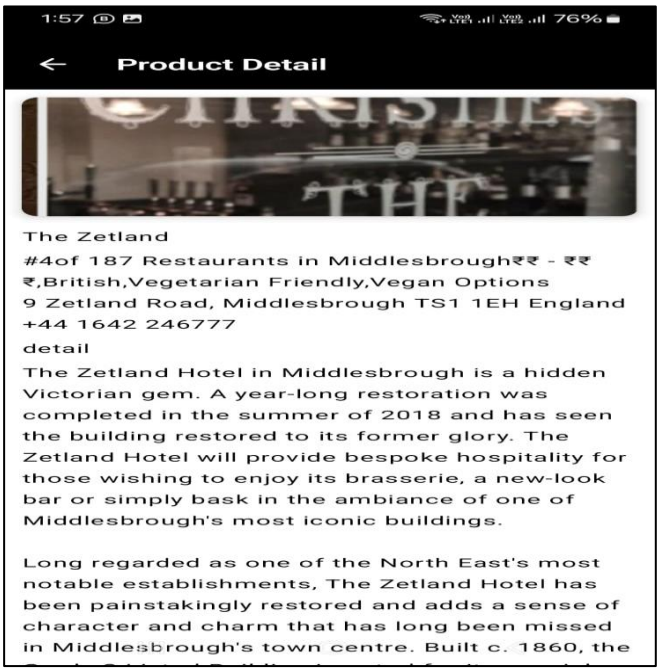


Figure 47: Details Page VI

6. Submit Booking –

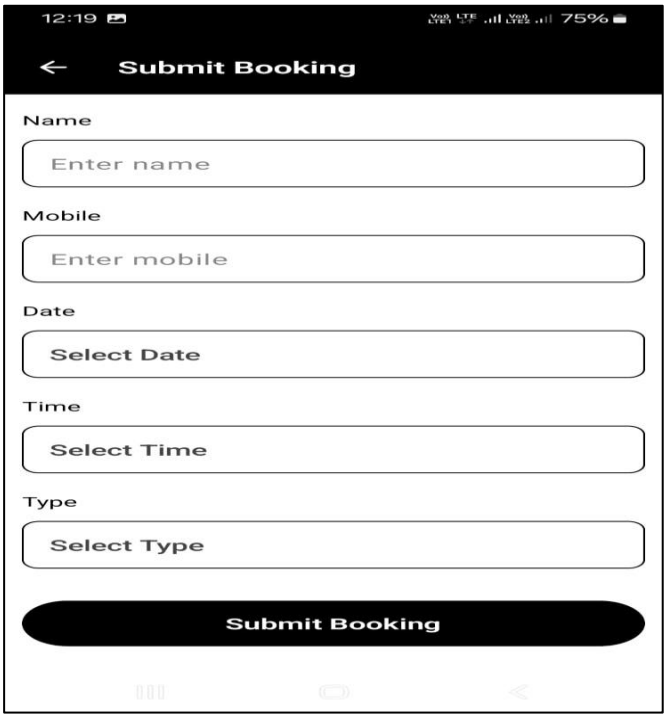


Figure 48: Submit Booking

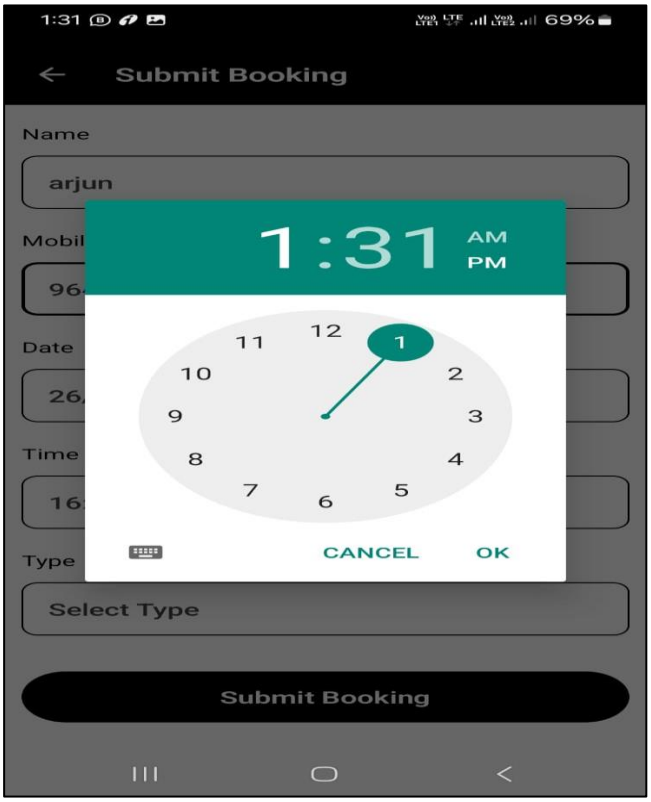


Figure 49: Submit Booking (Time)

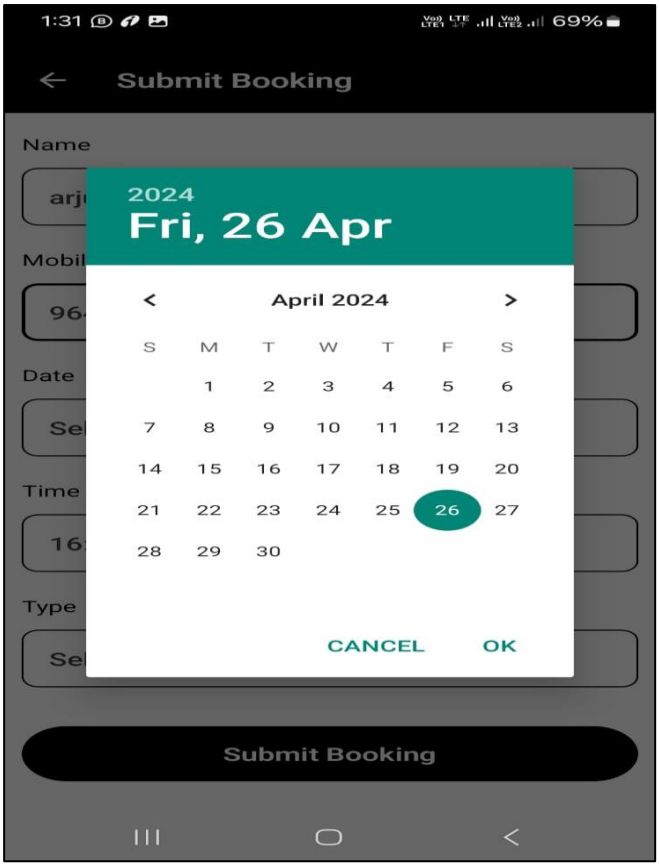


Figure 50: Submit Booking (Date)

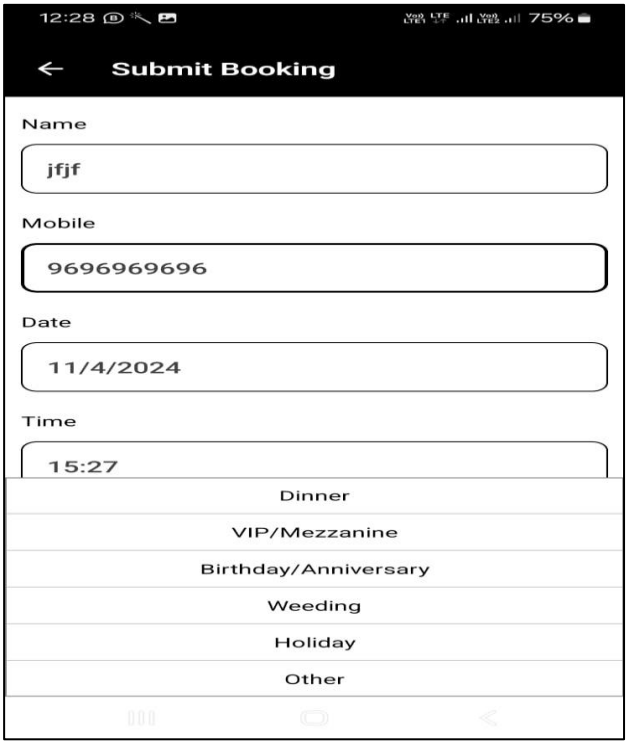


Figure 51: Submit Booking (Dropdown)

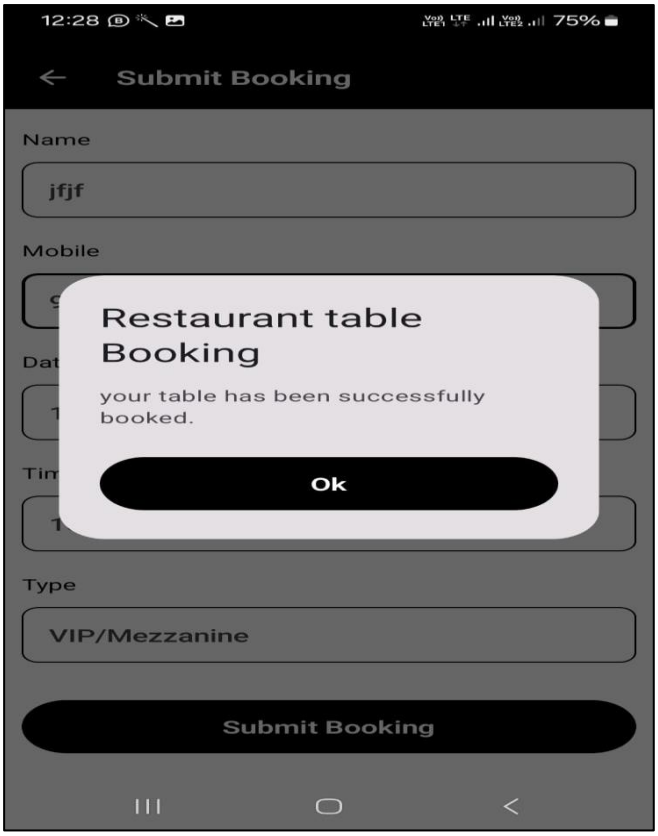


Figure 52: Booking Successful

7. Reviews and Ratings Page –

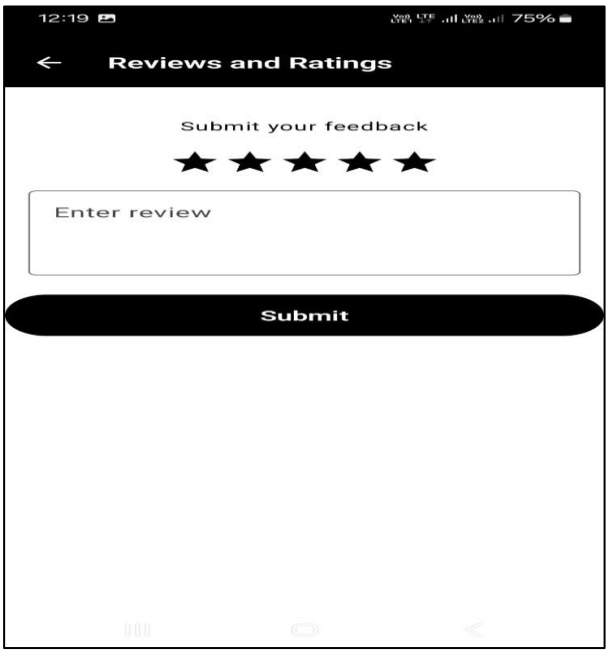


Figure 53: Reviews and Ratings

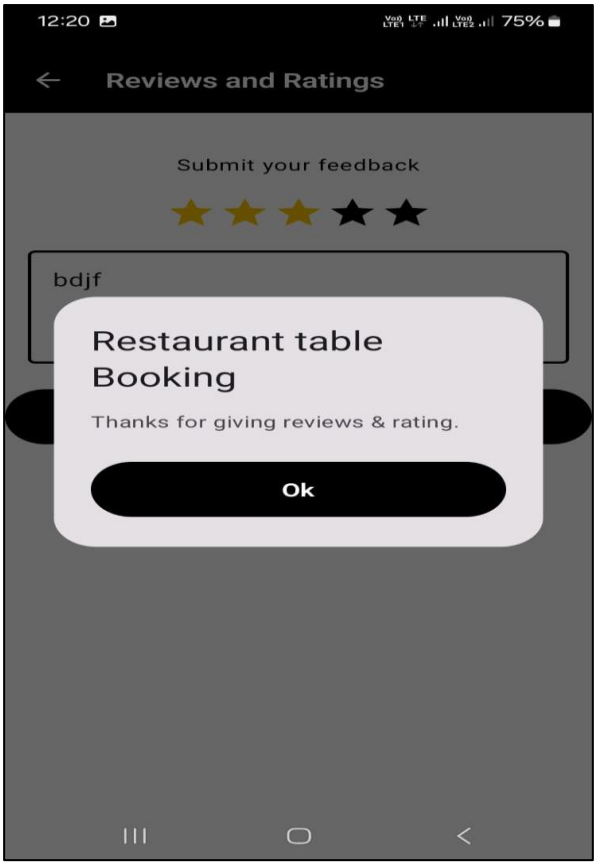


Figure 54: Submit Reviews

8. Log out Page –

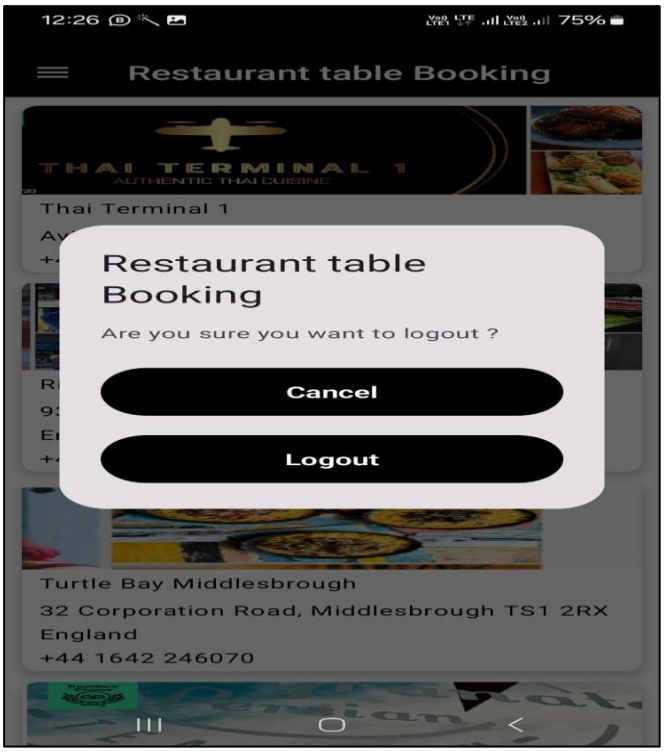


Figure 55: Log out Page

GIT

First Commit:

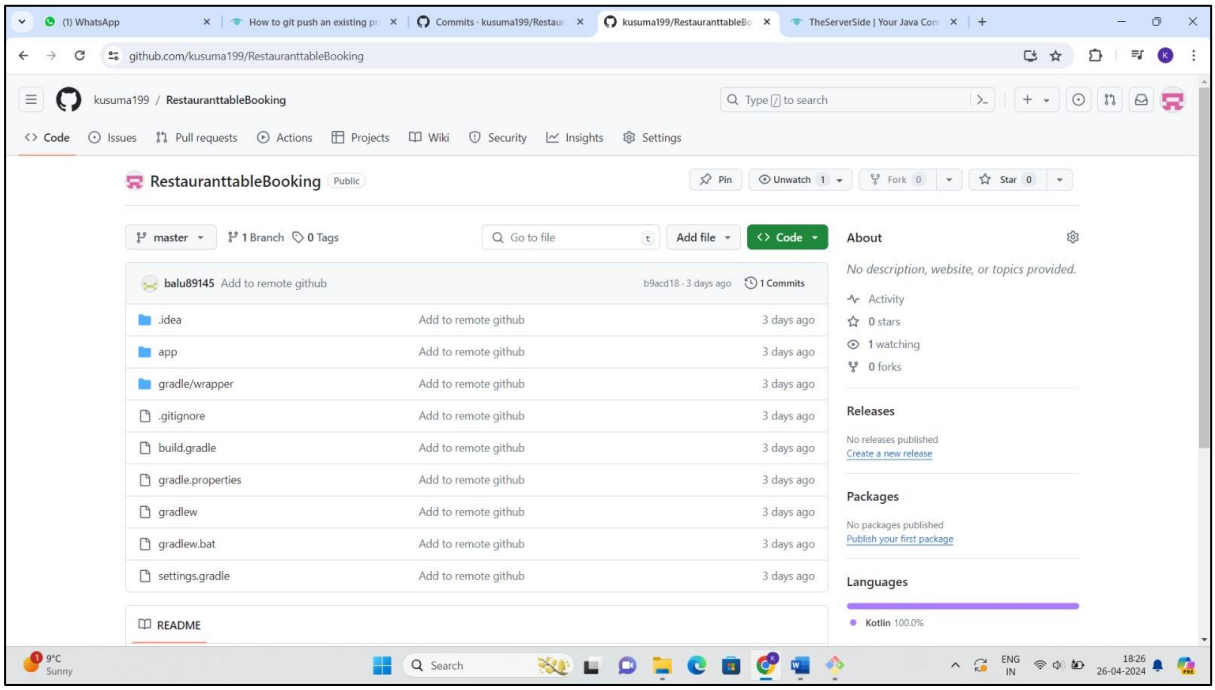


Figure 56: First Commit

Second Commit:

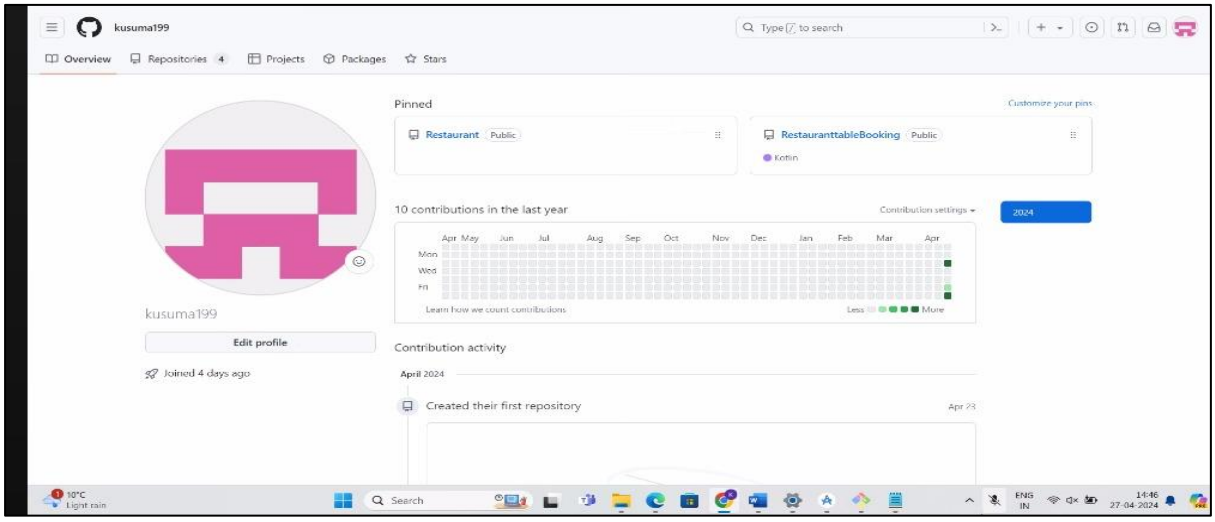


Figure 57: Second Commit

```
DELL@Kusuma MINGW64 ~/Downloads/second commit/clone/RestauranttableBooking (master)
$ git push
Enumerating objects: 76, done.
Counting objects: 100% (76/76), done.
Delta compression using up to 12 threads
Compressing objects: 100% (36/36), done.
Writing objects: 100% (45/45), 4.41 MiB | 2.31 MiB/s, done.
Total 45 (delta 14), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (14/14), completed with 14 local objects.
To https://github.com/kusuma199/RestauranttableBooking.git
   b9acd18..381efd5  master -> master

DELL@Kusuma MINGW64 ~/Downloads/second commit/clone/RestauranttableBooking (master)
$
```

```
DELL@Kusuma MINGW64 ~/Downloads/second commit/clone/RestauranttableBooking (master)
$ git commit -m "second commit"
[master 381efd5] second commit
 24 files changed, 432 insertions(+), 206 deletions(-)
 create mode 100644 app/src/main/java/com/restauranttablebooking/ui/feedback/Feedbackcreen.kt
 create mode 100644 app/src/main/res/drawable/ic_cluck.png
 create mode 100644 app/src/main/res/drawable/ic_glady.png
 create mode 100644 app/src/main/res/drawable/ic_pepper.png
 create mode 100644 app/src/main/res/drawable/ic_pome.png
 create mode 100644 app/src/main/res/drawable/ic_porto.png
 create mode 100644 app/src/main/res/drawable/ic_rio.png
 create mode 100644 app/src/main/res/drawable/ic_rose.png
 create mode 100644 app/src/main/res/drawable/ic_sandler.png
 create mode 100644 app/src/main/res/drawable/ic_thai.png
 create mode 100644 app/src/main/res/drawable/ic_turtle.png
 create mode 100644 app/src/main/res/drawable/ic_zetland.png

DELL@Kusuma MINGW64 ~/Downloads/second commit/clone/RestauranttableBooking (master)
$ git push
```

```
DELL@Kusuma MINGW64 ~/Downloads/second commit/clone/RestauranttableBooking (master)
$ git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app/build.gradle', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app/google-services.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app/proguard-rules.pro', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app/src/androidTest/java/com/restauranttablebooking/ExampleInstrumentedTest.kt', LF will be replaced
y CRLF the next time Git touches it
warning: in the working copy of 'app/src/main/java/com/restauranttablebooking/MainActivity.kt', LF will be replaced by CRLF the next
me Git touches it
warning: in the working copy of 'app/src/main/java/com/restauranttablebooking/ui/drawer/NavDrawer.kt', LF will be replaced by CRLF th
next time Git touches it
warning: in the working copy of 'app/src/main/java/com/restauranttablebooking/ui/model/RestaurantModel.kt', LF will be replaced by CR
 the next time Git touches it
warning: in the working copy of 'app/src/main/java/com/restauranttablebooking/ui/theme/Color.kt', LF will be replaced by CRLF the nex
time Git touches it
warning: in the working copy of 'app/src/main/java/com/restauranttablebooking/ui/theme/Theme.kt', LF will be replaced by CRLF the nex
time Git touches it
warning: in the working copy of 'app/src/main/java/com/restauranttablebooking/utils/RoundedButton.kt', LF will be replaced by CRLF th
next time Git touches it
warning: in the working copy of 'app/src/main/res/drawable-v24/ic_launcher_foreground.xml', LF will be replaced by CRLF the next time
it touches it
warning: in the working copy of 'app/src/main/res/drawable/ic_launcher_background.xml', LF will be replaced by CRLF the next time Git
ouches it
warning: in the working copy of 'app/src/main/res/mipmap-anydpi-v26/ic_launcher.xml', LF will be replaced by CRLF the next time Git t
ches it
warning: in the working copy of 'app/src/main/res/mipmap-anydpi-v26/ic_launcher_round.xml', LF will be replaced by CRLF the next time
it touches it
warning: in the working copy of 'app/src/main/res/mipmap-anydpi-v33/ic_launcher.xml', LF will be replaced by CRLF the next time Git t
ches it
warning: in the working copy of 'app/src/main/res/values/colors.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app/src/main/res/values/strings.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app/src/main/res/values/themes.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app/src/main/res/xml/backup_rules.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app/src/main/res/xml/data_extraction_rules.xml', LF will be replaced by CRLF the next time Git touch
 it
```

Figure 58: Ist Commit (master)

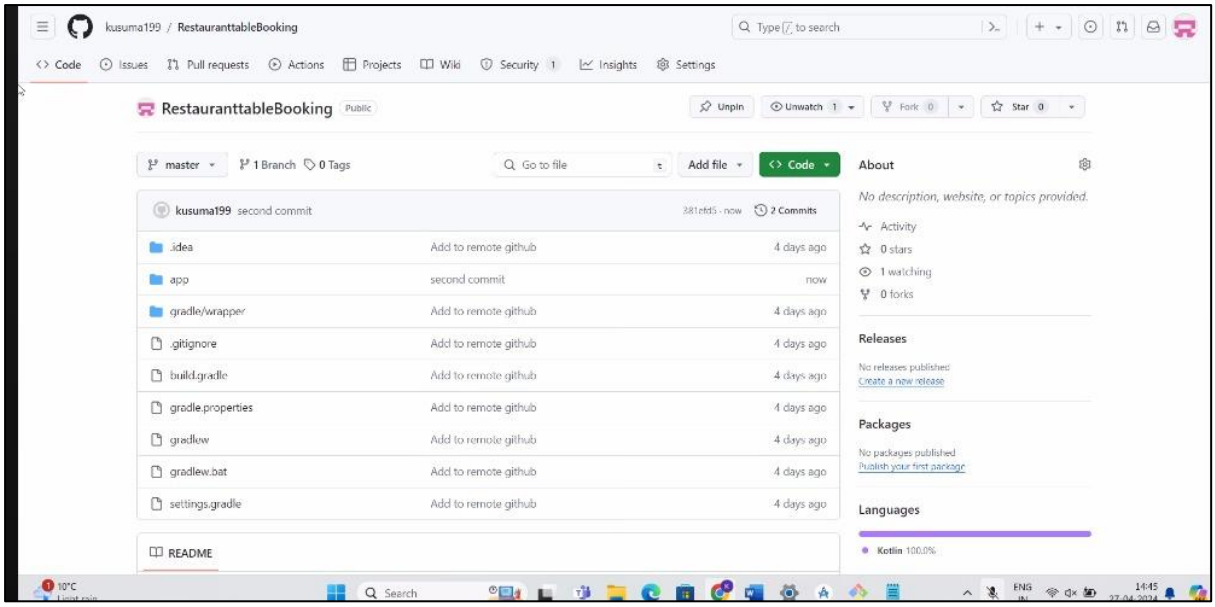


Figure 59: Ist Commit (Code)

Reference

A. Seymour, T., Hussain, J.Z. and Reynolds, S., 2014. How to create an app. *International Journal of Management & Information Systems (IJMIS)*, *18*(2), pp.123-138.

B. Hawley-Hague, H., Tacconi, C., Mellone, S., Martinez, E., Ford, C., Chiari, L., Helbostad, J. and Todd, C., 2020. Smartphone apps to support falls rehabilitation exercise: app development and usability and acceptability study. *JMIR mHealth and uHealth*, *8*(9), p.e15460.

C. Ellul, C., Gupta, S., Haklay, M.M. and Bryson, K., 2013. A platform for location based app development for citizen science and community mapping. *Progress in Location-Based Services*, pp.71-90.

D. Dekhane, S., Xu, X. and Tsoi, M.Y., 2013. Mobile app development to increase student engagement and problem solving skills. *Journal of Information Systems Education*, *24*(4), pp.299-308.

E. Kaur, A., 2018. App Review: Trello. *Journal of Hospital Librarianship*, *18*(1), pp.95-101.

F. Kaim, R., Härting, R.C. and Reichstein, C., 2019. Benefits of agile project management in an environment of increasing complexity—a transaction cost analysis. In *Intelligent Decision Technologies 2019: Proceedings of the 11th*

KES International Conference on Intelligent Decision Technologies (KES-IDT 2019), Volume 2 (pp. 195-204). Springer Singapore.

G. Gomes Silva, F.J., Kirytopoulos, K., Pinto Ferreira, L., Sá, J.C., Santos, G. and Cancela Nogueira, M.C., 2022. The three pillars of sustainability and agile project management: How do they influence each other. *Corporate Social Responsibility and Environmental Management*, *29*(5), pp.1495-1512.

H. Zasa, F.P., Patrucco, A. and Pellizzoni, E., 2020. Managing the hybrid organization: How can agile and traditional project management coexist?. *Research-Technology Management*, *64*(1), pp.54-63.

I. Collins, M.J., 2010. Burndown Charts. In *Pro Project Management with SharePoint 2010* (pp. 143-171). Berkeley, CA: Apress.

J. Manifesto, A., 2001. Manifesto for agile software development.