IT314 Software Engineering

Lab Session IV Specifying Tools and Technology

Project: Canteen Automation System

Prepared by

Team No. 31 | Group 6

Group Members

202001023 Vandan Patel
202001409 Kashyap Panchani
202001271 Dhruv Patel
202001415 Pinak Trivedi
202001403 Jaimin Baurasi
202001420 Divya Patel
202001405 Kenil Vaghasiya
202001430 Aryan Patel
202001407 Karan Patel
202001432 Aditya Jadeja
202001465 Jaykumar Navadiya

Under the guidance of

Prof. Khare, Prof. Tiwari and
Teaching Assistant Ankita Ma'am

Date

March 10, 2023

Index

1. Tools, Technologies and Frameworks

For developing mobile and web applications (iOS, Android, Web, MacOS and Windows):

→ Programming Language used for the purpose:

→ Framework used for developing the application:

→ Tool used for learning the Dart Language:

→ Debugging tools for Flutter & Dart Debugging:

Dart Devtools and VS Code inbuilt Debugger tool bundled with the Flutter and Dart Extention of VS Code.

→ Tools to be used for CI/CD of the application:

Codemagic for flutter CI/CD
or

GitHub Actions

Github Actions

→ Command line tool for installation purposes:

pub.dev

→ Main Packages to be used for iOS Front-end:

Cuppertino_icons and widget packages

→ Main Packages to be used for Android Front-end:

Material Theme package for Android like native front-end.

→ Main Packages for Back-end connection (Database and Others):

1. elastic_client package: Dart bindings for ElasticSearch HTTP API. ElasticSearch is a full-text search engine based on Lucene.
2. url_launcher: Flutter plugin for launching a URL. Supports web, phone, SMS, and email schemes.
3. http: A composable, multi-platform, Future-based API for HTTP requests.
4. firebase_core: Flutter plugin for Firebase Core, enabling connecting to multiple Firebase apps.
5. firebase_auth: Flutter plugin for Firebase Auth, enabling Android and iOS authentication using passwords, phone numbers and identity providers like Google, Facebook and Twitter.
6. firebase_messaging: Flutter plugin for Firebase Cloud Messaging, a cross-platform messaging solution that lets you reliably deliver messages on Android and iOS.

→ IDE to be used for the development of the Flutter Application:

→ Autosuggestion tool for auto code completion for the Dart Programming Language:

2. Selection of Database

→ Firebase [ ]:
For main database purposes the integration is easy with the Flutter framework and is a No-SQL Database. Also, it is hosted on the google cloud, so on-premise resources are not required.

→ Elasticsearch [elastic]
We plan to use the Elasticsearch service for implementing the flutter search box (to search anything on the application). We will see if the firebase database could be passed on/ replicated on the elasticsearch service/database as it is not reliable as of now for using with Flutter framework due to lack of proper documentation of elastic_client package.

3. Estimating effort of the project - Function point analysis for user stories

User Story 1:-

As a customer, I want to log in securely using the app, so that I can protect my account and personal information.

User Story 2:-

As a customer, I want to order food using the app, so that I can save time and avoid queues.

User Story 3:-

As a customer, I want to check my order status using the app, so that I can know when my food is ready for pickup.

User Story 4:-

As a customer, I want to receive notifications from the app, so that I can be alerted when my food is ready for pickup or when there are any changes or issues with my order.

User Story 5:-

As a customer, I want to show my unique QR code when picking up my order, so that I can verify my identity and prevent unauthorized access to my food.

User Story 6:-

- As a customer, I want to pay for my food using the app through the integrated payment system, so that I can complete my order quickly and securely.

User Story 7:-

- As a customer, I want to see recommendations for similar items and items I might like based on my preferences and previous orders, so that I can discover new and better food options for me and order faster.

User Story 8:-

As a customer, I want to give feedback on the food I ordered and the cafeteria I ordered from, so that cafeteria owners can improve their quality of service and other users can make informed decisions about the food they want to order.

User Story 9:-

As a cafeteria owner, I want to have a secure and separate login system for my account, so that I can prevent unauthorized access to my personal information and manage my cafeteria online.

User Story 10:

As a cafeteria owner, I want to have an easy-to-use interface and advanced features for managing my business, so that I can save time and increase efficiency.

User Story 11:

As a cafeteria owner, I want to have an automated payment system for accepting payments from customers, so that I can reduce errors and increase convenience.

User Story 12:

As a cafeteria owner, I want to have a system that can let me change the menu and prices of my food items easily and quickly, so that I can adapt to customer demand and market changes.

User Story 13:

As a cafeteria owner, I want to create special offers for my customers, so that I can increase sales and loyalty.

External Outputs:

1. Confirmation message indicating successful login
2. Error message indicating invalid login credentials or account lockout
3. Order confirmation message indicating that the order has been received
4. Order summary and estimated delivery time
5. In-app or email receipt for the order
6. Push notifications for updates on the order status (if applicable)
7. Order status message indicating the current status of the order (e.g., being prepared, ready for pickup)
8. In-app or email notification for updates on the order status
9. Push notifications for updates on the order status (if applicable)
10. In-app notification alerts
11. Email or SMS notifications (if specified by the customer)
12. Unique QR code displayed on the customer's device
13. Confirmation message indicating successful verification of the QR code
14. Confirmation message indicating successful payment
15. In-app or email receipt for the payment
16. Error message indicating a failed payment or invalid payment information
17. Recommended menu items based on customer preferences and previous orders
18. In-app or email notifications for new recommendations
19. Confirmation message indicating successful submission of feedback
20. In-app or email notification for updates on feedback status, account activity and security
21. Updated cafeteria information (e.g., menu items, prices, hours)
22. Analytics and reports on cafeteria activity and performance
23. Error messages indicating a failure to complete tasks or access certain features
24. Confirmation message indicating successful payment
25. Error messages indicating payment failure or other issues
26. In-app or email receipts for customers
27. Updated menu items and prices displayed on the customer-facing app or website
28. Confirmation message indicating successful menu and pricing changes
29. Error messages indicating failed attempts to make changes
30. Display of special offers on the customer-facing app or website
31. Confirmation message indicating successful creation of special offers
32. Error messages indicating failed attempts to create special offers

External Inputs:

1. User-entered login credentials (username and password)
2. Menu items and quantities selected by the customer
3. Delivery or pickup preferences (if applicable)
4. Customer identification information (e.g., order number, phone number)

5. Customer preferences (e.g., dietary restrictions, favorite cuisine)
6. Customer feedback (e.g., rating, comments, suggestions)
7. Order details (e.g., order number, food item, cafeteria name)
8. Cafeteria owner login credentials (e.g., email address, password)
9. Cafeteria information (e.g., menu items, prices, hours)
10. Customer payment information (e.g., credit card, debit card, digital wallet)
11. New menu item information (e.g., name, description, price)
12. Updated pricing information
13. Special offer information (e.g., name, description, discount percentage or amount, expiration date)

External Inquiries:

1. Retrieving account information (e.g., balance, transaction history) after login
2. Checking the status of an order
3. Requesting changes or cancellations to an order
4. Reporting issues or problems with an order
5. Customer identification information (e.g., order number, phone number)
6. Requesting support for login issues
7. Checking payment history
8. Retrieving customer preferences and order history
9. Requesting feedback on recommended items
10. Requesting support for technical issues/questions about using advanced features
11. Checking sales or inventory reports
12. Requesting updates on customer feedback or complaints
13. Requesting feedback on specific food items or cafeterias
14. Requesting support for technical issues with the payment system
15. Checking menu and pricing history
16. Requesting updates on changes or issues with the system.

External Interface Files:

1. QR codes of all orders placed: These files are used to store images, videos, or other multimedia content that is used by the software application. For example, a video editing software application may use multimedia files to import and export video content.
2. UPI APIs: These are interfaces that allow the software application to communicate with external systems through the internet. Http package in Dart from pub.dev.

Internal Logic Files:

1. Login credentials
2. Order information (date, time, items, quantity, status, etc.)

3. Customer order history
4. Customer dietary preferences and taste profile database
5. Item database with relevant attributes such as price, ingredients, nutrition facts, and availability
6. Database of food items with attributes such as taste, quality, quantity, price, and service
7. Database of cafeterias with attributes such as cleanliness, ambiance, staff friendliness, and convenience
8. Database of customer feedback (ratings and optional text review) with attributes such as item/cafeteria ID, customer ID, rating category, rating score, and text review
9. Menu item file
10. Sales data file
11. Inventory and supply chain file
12. Payment information file

Complexity Factor Analysis:

Complexity factor: Fi	Value = 0	Value = 1	Value = 2	Value = 3	Value = 4	Value = 5	Fi
Performance	0	0	0	0	1	0	4
Distributed Processing Functions	0	0	0	1	0	0	3
Online Database Management	0	0	0	0	0	1	5
Reporting and Analytics	0	0	0	1	0	0	3
Complexity of Processing	0	0	0	0	1	0	4
Multi-Platform Installations	0	0	0	0	0	1	5
Code Maintainability	0	0	0	1	0	0	3
Security	0	0	0	0	1	0	4
User Interface	0	0	0	0	0	1	5
						ΣFi =	36

Complexity Adjustment Factor(CAF) = 0.65 + 0.01*36 = 1.01

Measurement Parameters:

Measurement Paremeters	Low	Avg	High
External Inputs (El)	3	4	6
External Output (EO)	4	5	7
ExternalInquiries (EQ)	3	4	6
Internal Logical Files (ILF)	7	10	15
External Interface Files (EIF)	5	7	10

Unadjusted Function Count:

Count of Measurement parameters	Low	UFP for Low Category	Avg	UFP for Avg Category	High	UFP for High Category
External Inputs (EI)	**8**	**24**	**4**	**16**	**1**	**6**
External Output (EO)	**24**	**96**	**6**	**30**	**2**	**14**
External Inquiries (EQ)	**10**	**30**	**4**	**16**	**2**	**12**
Internal Logical Files (ILF)	**8**	**56**	**3**	**30**	**1**	**15**
External Interface Files (EIF)	**1**	**5**	**1**	**7**	**0**	**0**
Totals for categories		**211**		**99**		**47**
						UFC = 357

FP = UFC * CAF = 357 * 1.01 = 360.57

Effort = FP * Productivity = 360.57 * 2.5 = 901.425 Person Hours.