Team Oriented Project

Building Kitchen Module

The Smart Office app is intended to provide employees of eXXcellent solutions GmbH with convenient access to functionalities that are needed for working in the office. This app focuses on food ordering functionality, aimed at simplifying and streamlining the meal preparation process.

Project Manager / Scrum Master:	Yatharth Upmanue
Members:	Sophia Soares Silva Savo Simeunović Marin Sušić Vasiliy Klyosov Yatharth Upmanue
Budget:	€ 0
Timeline:	04/10/2023 - 28/11/2023

Table of Contents

Introduction 4
Motivation and Project Goal 5
Requirements 6
Functional Requirements 6
Non-functional Requirements 6
Technical Concept 8
External services 9
Interaction of important model Classes 13
Project Management 18
Team Organization 18
Timeline of the project 19
Velocity Report 21
Sprints 21
Member Contribution 25
Sophia Soares Silva 25
Savo Simeunovic 25
Vasiliy Klyosov 26
Marin Susic 27
Yatharth Upmanue 27
Result 29
Achievements and Limitations 39
Customer Feedback 39
Lessons Learned 40
Outlook and Further developments 42
Feature Roadmap 42
Feature 1: Redirecting the Chef from Statistics to Order Review 42
Feature 2: Redirecting the Chef from Wish List to Meal Creation 42
Feature 3: Chef Can Add Surprise Dishes for Employees 44
Feature 4: AI-Generated Recipes for Chef Meals and Employee Viewing 45
Technology Upgrades 46
Performance enhancements 47
Quality assurance 47
Administrator Implementation 47
Design and Usability Enhancements 48

Security improvement 48
Compliance and Regulatory updates 49
References 51
Appendix 52
I. Preliminary Design to the customer 52
II. Example of our Sprint Review 52
III. Installation guide 52
IV. Poster – Version 1 and 2 52
V. Presentation 52

Introduction

"Welcome to the exciting new project aimed at revolutionizing the way the employees order breakfast and lunch at eXXcellent solutions GmbH! In this student-led initiative, we embarked on the development of a kitchen module in their already existing app using cutting-edge Flutter technology that not only simplifies the ordering process but also introduces a range of enhanced **'kitchen features'** to optimize our culinary experience.

The new application will empower our colleagues to effortlessly place meal orders, offering a selection of dishes that can be pre-ordered for the upcoming 4-6 weeks. Additionally, the app module will facilitate seamless dish management, allowing the kitchen staff to add, update and delete menu items with ease.

One of the most innovative aspects of the app is its automated communication with the kitchen. Through integration with an email, the kitchen staff will receive updates on the quantity of orders for each dish and at what time the dish has to be served, at least one day in advance.

Furthermore, the app will utilize **Artificial Intelligence** to generate images for the daily dishes, providing an intriguing opportunity to explore the accuracy of AI-generated food images.

In addition to these features, the app will offer statistical providing valuable insights into eating habits and dish popularity. Moreover, a unique 'Buffet Alarm Buzzer' feature will help reduce food waste by allowing the kitchen staff to notify colleagues of available food, even if they haven't placed an order for the day.

By implementing this project, we aim to significantly enhance the efficiency and user-friendliness of breakfast and lunch ordering process for eXXcellent solutions GmbH.

Motivation and Project Goal

The project was initiated to address the inefficiencies and limitations of the current system for ordering lunch and breakfast at eXXcellent solutions. The existing process is cumbersome, lacking in user-friendliness, and is not fully utilizing technology to streamline operations. The project aims to solve these issues by introducing a user-friendly Flutter app that not only simplifies the ordering process but also incorporates additional "kitchen features" to enhance the overall experience.

The opportunity this project capitalizes on is the potential to significantly improve the efficiency and user-friendliness of breakfast and lunch ordering. This project aims to revolutionize the dining experience for colleagues at eXXcellent solutions by leveraging technology and incorporating features such as pre-ordering, dish management, automated communication with the kitchen, AI-generated food images, statistical functions, a "Buffet Alarm Buzzer", multiple Language support, notifications and a lot more features which will be covered later on in the Documentation.

SMART Goals:

Specific: The project aims to develop a user-friendly Flutter app with enhanced kitchen features to streamline the breakfast and lunch ordering process.

Measurable: The success of the project will be measured by the increase in efficiency, user satisfaction, and the reduction of food waste.

Achievable: The project's goals are realistic and attainable within the scope of the resources and expertise available.

Relevant: The project directly addresses the need for a more efficient and user-friendly food ordering system at eXXcellent solutions.

Time-bound: The project has a clear timeline for development and implementation, with specific milestones and deadlines for each phase.

Requirements

Functional Requirements

- As a user, I want to be able to place meal orders for breakfast and lunch in advance for the next 4-6 weeks, so that I can plan my meals ahead and ensure availability of my preferred dishes.
- As a user, I want to be able to rate and leave feedback for the meals that I ate.
- As a user, I want the system to have capability to invite guests and place order for them simultaneously.
- As a user, I want to receive regular notifications about any changes that incur with regards to my meal that I have ordered. I would also like the system to remind me an hour before the expiration, if I would like to order my food for the next day, if not already done so.
- As a kitchen staff member (Chef), I want to receive automated updates on the quantity of orders for each dish and the serving time at least one day in advance, so that I can prepare and serve meals efficiently.
- As a kitchen staff member (Chef), I want the system to provide meal rating and feedback so that I can make inform myself of people's likes and dislikes.
- As a kitchen staff member (Chef), I want to send out a notification of meal being ready to the users who have ordered a meal for that particular day

These user stories encompass the specific behaviors and functionalities that the system will have, addressing the needs and benefits for different types of users within the eXXcellent solutions GmbH environment.

Non-functional Requirements

- As a user I want to have an allergy list for the meals that are being prepared.
- As a user, I want the ability to be able to add an allergy if it is not present in the standard list.
- As a user, I want to be able to select the Language of my choice.
- As a user, I want to be able to switch easily from Light mode to Dark mode and vice versa, depending on what I want to use on a particular day.
- As a user, I want to be able to provide to the kitchen staff member (Chef), with more ideas of what I would like to be cooked. (Wishlist)

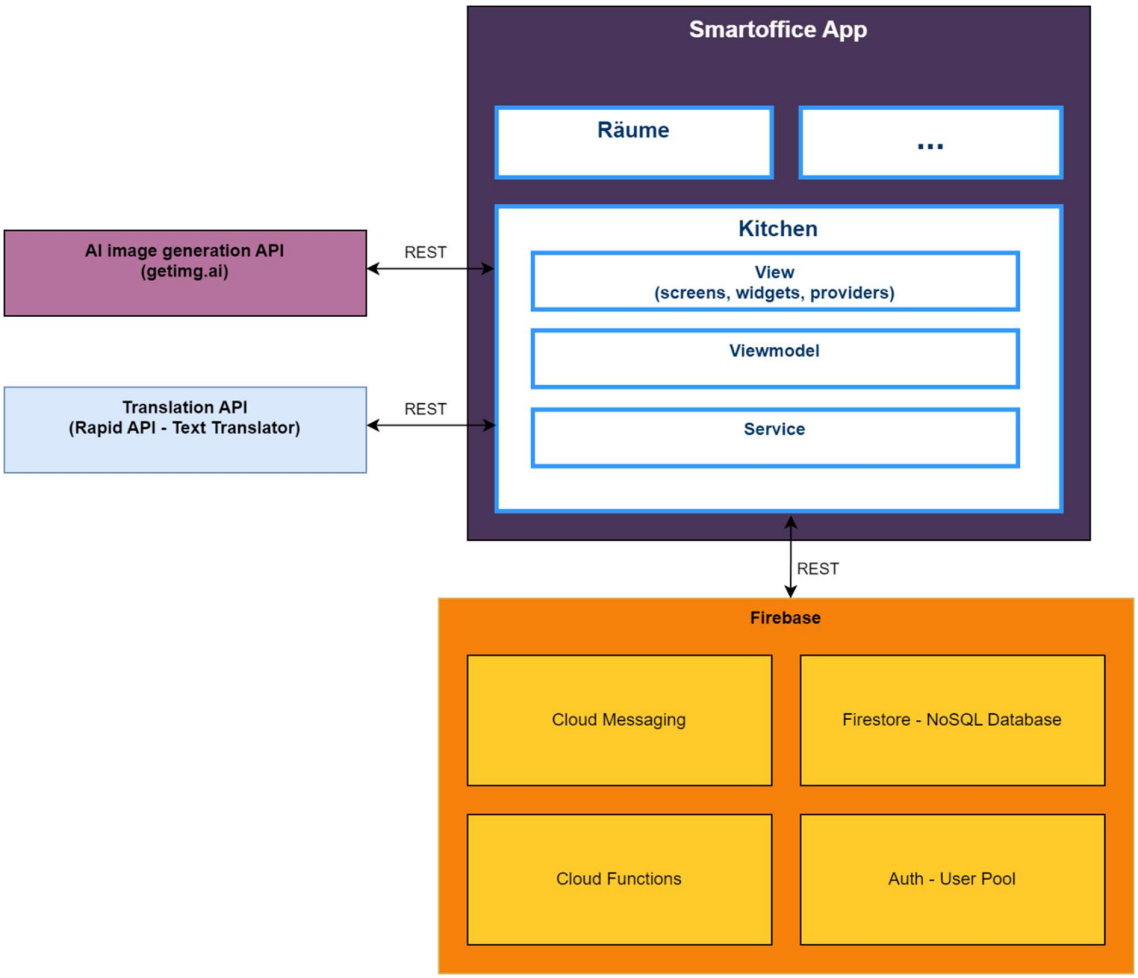
- As a kitchen staff member (Chef), I want to easily add, update, and delete menu items, so that I can efficiently manage the menu and accommodate changes in dishes.
- As a kitchen staff member (Chef), I want the system to have a "Buffet Alarm Buzzer" feature to notify colleagues of available food, even if they haven't placed an order for the day, so that we can reduce food waste and ensure food is utilized efficiently.
- As a kitchen staff member (Chef), I want the system to provide statistical insights into eating habits and dish popularity, so that I can make informed meal choices.
- As a Kitchen staff member (Chef), I want the system to utilize Artificial Intelligence to generate images for the daily dishes, so that the users can visually have an idea what is on the menu for the day.
- As a Kitchen staff member (Chef), I want the system to utilize Artificial Intelligence to generate images for the daily dishes, so that the users can visually have an idea what is on the menu for the day.

These user stories encompass the specific behaviors and functionalities that the system will have, addressing the extra optional features and benefits for different types of users within the eXXcellent solutions GmbH environment.

Technical Concept

High-level explanation of the App

As previously mentioned, this application is build on top of already developed application. It is only one component which can be seen in the picture bellow. The structure of our component is the same as the other components. View part contains everything about what user sees (UI) and providers of the screens. Viewmodel can be seen as the models which are used to contain information received from the database and to display it in the UI. Service contains classes used to communicate with outside of the application (different APIs, Firebase…)



Smartoffice App – Kitchen component

External services

AI image generation API

The AI used to generate images of each Meal is called getimg.ai. The website can be found here. With the free plan, each account has 100 free images per month. The only thing that has to be changed in order to change to some other account is the API key. The prompts are just strings that combine main dish, side dishes and some strings that enhance the image quality. For more information on how to use this API, please refer to the documentation provided by the website.

Translation API

AI image generation API understands only the prompts written in English language and the Chef that would create meals only understands German so we need a way to solve this problem. This is why we use this API, built by RapidAPI, to translate from German to English language. Free plan on this website provides 100000 translated characters for free.

Cloud Functions

Cloud Functions are function that are executed anytime some condition is met. In our case, we used them to trigger the notifications every Sunday at 9:30 to notify the Chef about the number of ordered breakfasts and every Monday and Wednesday at 12:30 to notify the Chef about the number of ordered lunches (1 day in advance). For scheduling, the Google Cloud service called Cloud Scheduler is used. Emails are send by using Nodemailer module of Node.js.

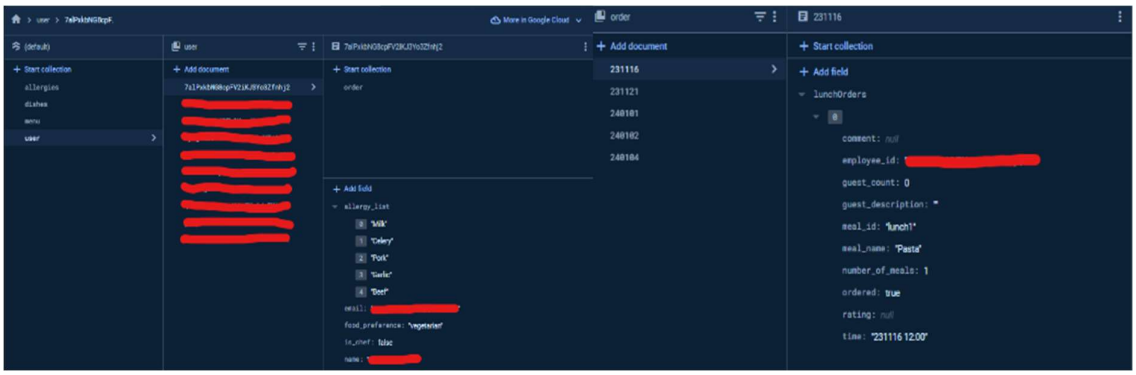
Firebase

Firebase are cloud services created by Google that are on top of the Google Cloud. We use only a few services which make our life easier.

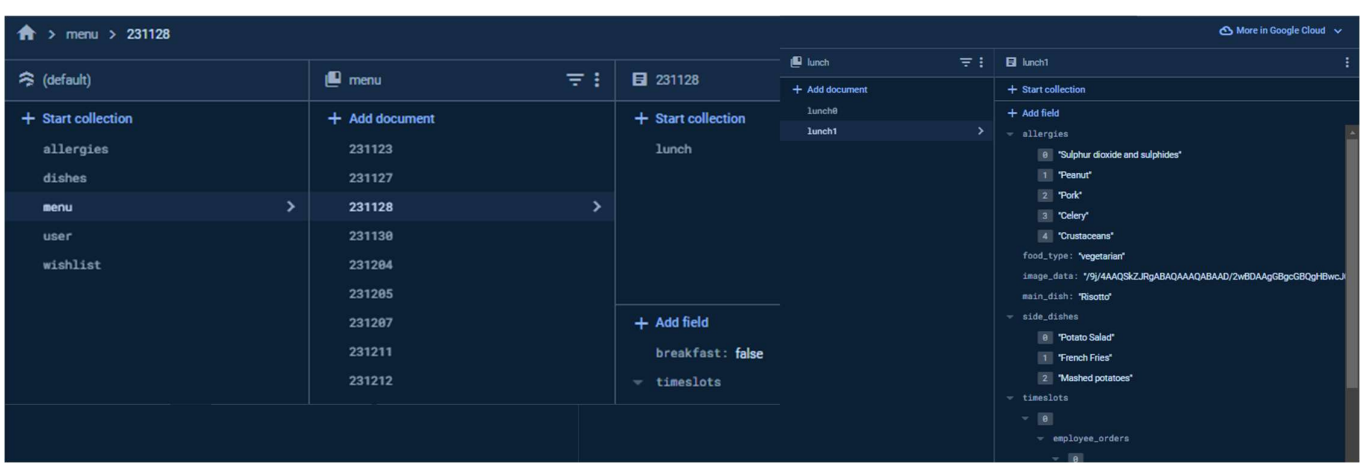
For creation of the user pool, we used Firebase Auth service. This makes creating users very simple and is easily implemented in any Flutter project. Whenever a new user is created in our app, an id is created for that user and that is used in Firestore database. Email confirmation and password reset are provided by this service and our job is only to call Auth functions, the rest is done by Firebase.

To store data, we use NoSQL database called Firestore. For details about how Firestore works, please refer to their detailed documentation. Here will only our structure of Firestore be explained

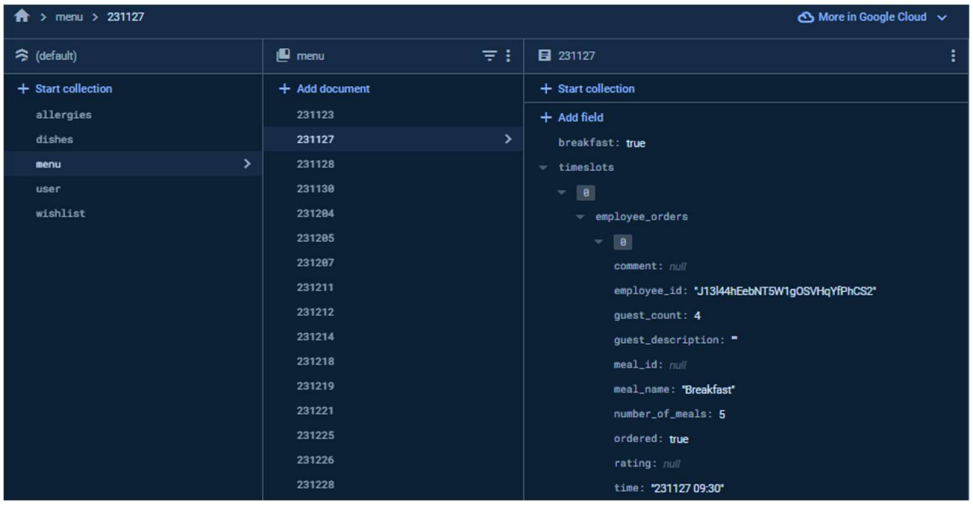
- **user** collection: Contains a list of documents where each document id is an id of users. Inside of each document, there is information such as allergies or food preference. Together with this, each one of those documents contains a collection called 'order' that contains a list of documents where id is the menu id (this id will be explained in **menu** collection). Inside of each one of those documents, there is an attribute that contains a list of orders for that menu of that specific user.



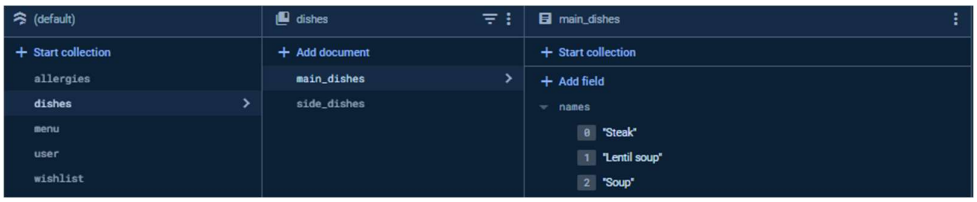
- **menu** collection: Contains a list of documents where each document id is made from the date that that menu is made for. It can be either breakfast or lunch menu. In case it is a lunch menu, there is a Boolean value indicating that, and in that case, there is a collection called 'lunch' that contains a list of meal for that menu and each meal contains all the information about that meal including a list of timeslots and a list of orders for each timeslot. The reason why we haven't used references for the dishes is that, in case some dish changes, we would change the past menus which we wanted to avoid since past can't be changed.



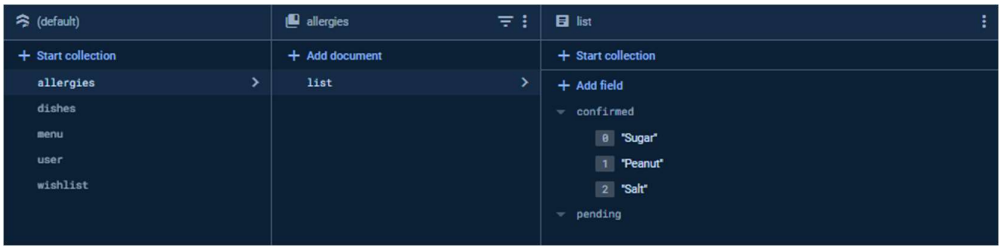
- In case it is a breakfast menu, the menu document only contains a list of timeslots containing all the orders:



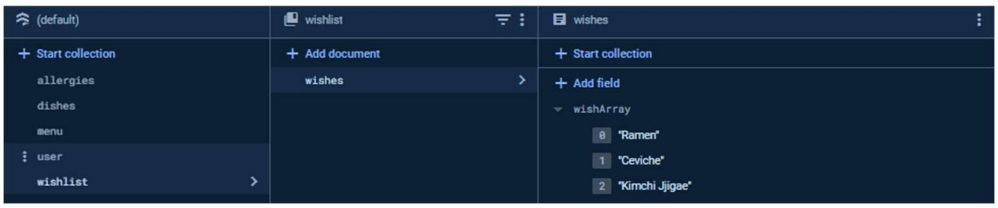
- **dishes**: Contains a list of side and main dishes that the Chef has previously created.



- **allergies**: Contains a list of pending allergies submitted by employees and a list of confirmed allergies confirmed by the Chef. When allergy is confirmed, the string is moved from pending list to the confirmed list. If any allergy is denied, the string gets removed from the lists.



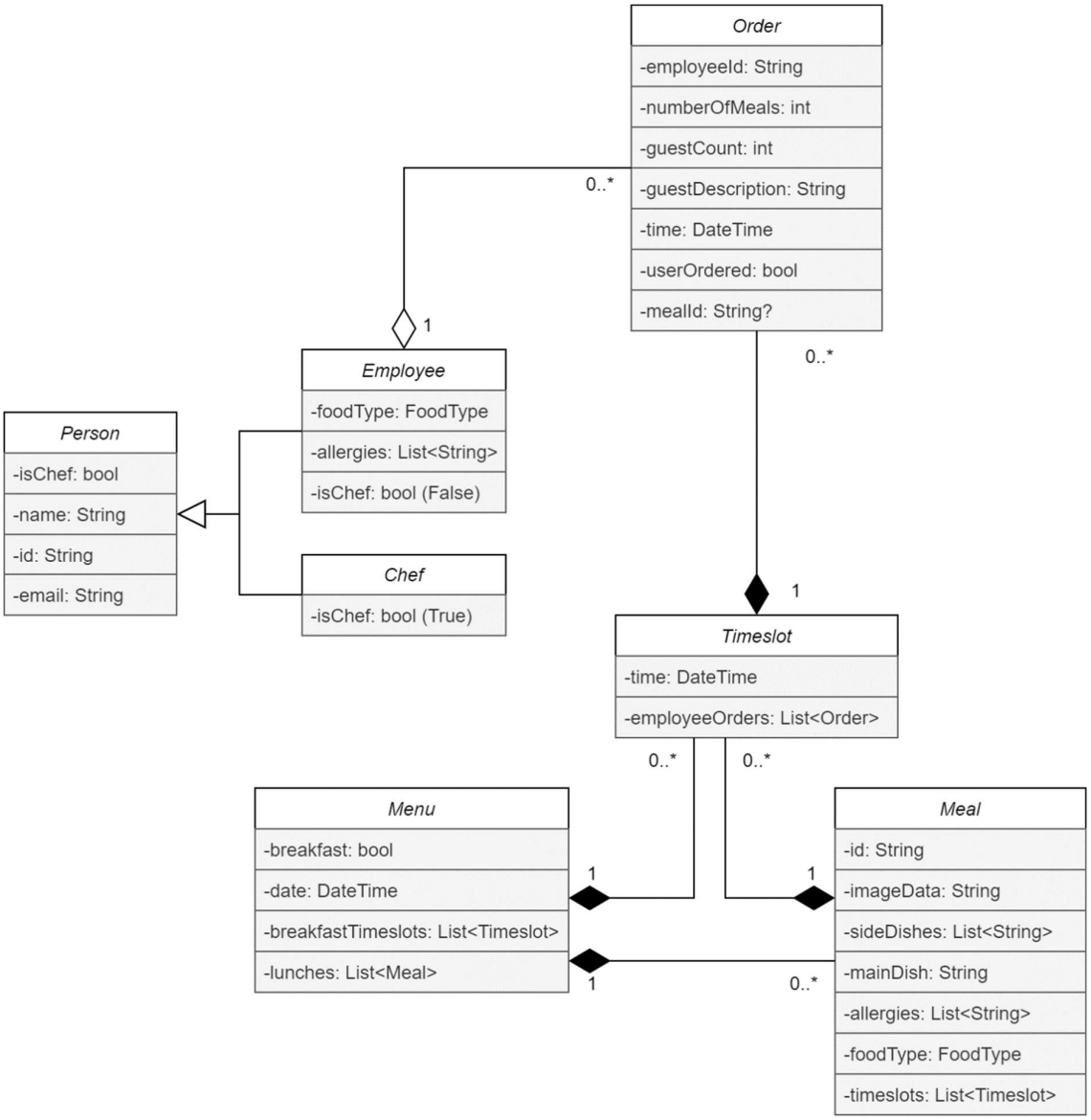
- **wishlist**: Contains a list of strings that represent the dishes that employees want Chef to make in the future.



Any call towards the Firestore or any other Firebase service is done through ‘service’ part of the application.

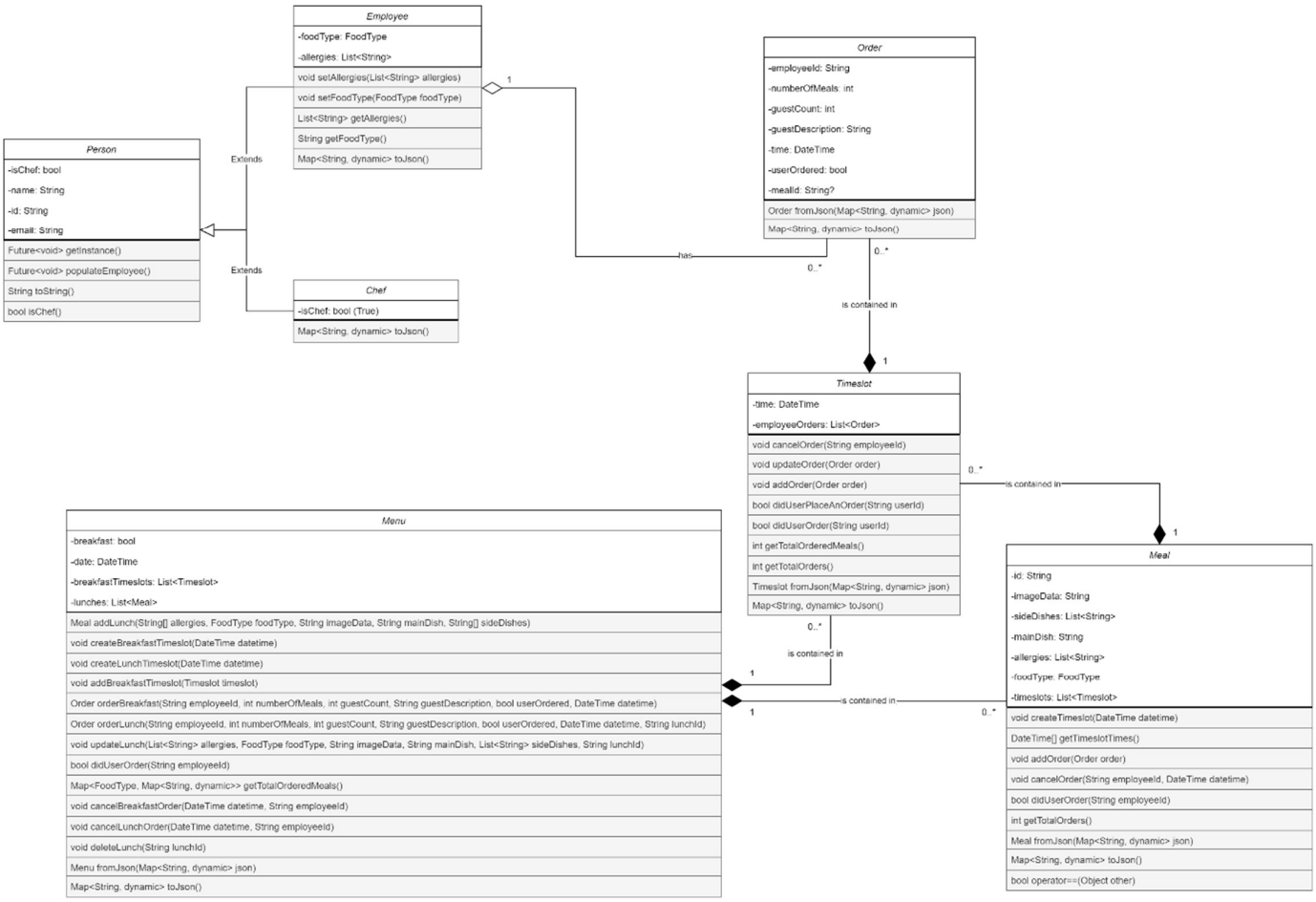
Interaction of important model Classes

To better understand how the models, interact inside of the application, take a look at the UML Class Diagram. It is a diagram that describes the structure of a system by showing the ystem's classes, their attributes and the relationships among them.



UML Class Diagram – simplified

In the diagram above, the simplified version of the Class diagram is shown. It does not contain any function of the classes. In the diagram bellow, all available functions are included for the detailed view.



UML Class Diagram - extended

User types

There are two different user types in the application: Employee and Chef. Both of these are inherited from an abstract singleton class Person. To access the singleton class and get the Employee or Chef class, the *getInstance* function has to be called. To differentiate between these two user types, there is a Boolean value inside of the database that is called *is_chef*. Each user has attributes *isChef*, *name*, *id* and *email*. In addition to that, each Employee has *foodType* and *allergies* which every user has to specify when registering. **To select which user is Chef, the only thing which has to be set manually is *is_chef* value inside of the database.** Based on the *isChef* attribute, user will be displayed different views.

Menu class

Menu is a class that acts as a container and management class of all meals and orders. Chef is able to create 2 different types of Menus for each day in the week: breakfast and lunch menu. **It is not possible, with the current structure, to have both on the same day**. When creating a new menu, the Date of the menu has to be specified and is it a breakfast menu or not. If Chef selects Monday, he is only be able to create a breakfast menu (currently, this is hard-coded in multiple places). In case of Tuesday or Thursday, Chef is able to create a lunch menu. (can be easily changed in only one place).

- **Lunch menu**: In case the lunch menu is being created, the class will contain only a list of Meal objects inside of the attribute *lunches*. Any manipulation of meals is done through the Menu object. It is hard-coded so that each Menu can only contain two Meals where one has to be vegetarian and the other one has to be the meat one (according to customer requirements to limit what Chef can do). Meals are given in multiple timeslots (for example, 12:00 and 12:30 for the customer) and because of that, each Meal has a list of Timeslot objects. These timeslots should be added through the Menu object to keep the consistency across all meals (so all meals have the same timeslots). Inside of each Timeslot object, there is a list of Order objects that contain information about each employee order places for that specific timeslot and that specific meal.

```
menu.addLunch(allergies: restrictions, imageData: imageData, ...);
menu.addLunch(allergies: restrictions, imageData: imageData, ...);
menu.createLunchTimeslot(datetime: DateTime(menu.getDate().year,
  menu.getDate().month, menu.getDate().day, 12, 30));
menu.createLunchTimeslot(datetime: DateTime(menu.getDate().year,
  menu.getDate().month, menu.getDate().day, 12, 00));

await FirestoreService().createNewMenu(menu);
```

- **Breakfast menu**: In case Chef chooses Monday in the calendar, he/she will create a breakfast menu. Because breakfasts don't have any information about them (no image, no dishes, no food type…), menu, in that case, contains an attribute *breakfastTimeslots* and meals are excluded. There is an argument that it would be

more flexible if in this case there were also Meal objects with null attributes but we decided not to do that. In the customer's case, breakfast is offered in only one timeslot (9:30) and it is hard-coded in providers but there is an option to include more than one. In the similar way to lunch menu, each timeslot contains a list of employee orders and any manipulation should be done through the Menu class.

```
menu.createBreakfastTimeslot(datetime: DateTime(menu.getDate().year,
  menu.getDate().month, menu.getDate().day, 9, 30));

await FirestoreService().createNewMenu(menu);
```

Meal class

Meal is a class that is used to store all information about each lunch. That includes data such as hash value of an image, a list of side dishes, a main dish, list of allergies of the meal, food type (can be vegetarian or non-vegetarian) and an id of the meal. Id is generated inside of the Menu class when a function *addLunch* is called. Since we are hard-coding that there should be only two meals (one for each food type), we are also hard-coding the ids. Non-vegetarian meal always have an id of 'lunch0' and vegetarian meal always have an id of 'lunch1'. All meals should have the same timeslots and this is why the timeslots are created from the Menu class by using *createLunchTimeslot* function. This ensures that each lunch inside of the list has the same timeslots and each newly created meal has the same timeslots as all meals which are already contained inside of the list.

Timeslot class

Timeslot class contains information at what time is the meal offered and a list of Order objects which contain information about all employee orders. Each Meal object contains a list of Timeslot objects which gives us a lot of flexibility.

Order class

This class is a container for all information about each order. It contains data such as:

- employeeId: The id of the user that places the order
- userOrdered: Since each user can invite guests for all other timeslots and meals, there could be multiple of orders for one employee. This Boolean value ensures that we

know which order was placed specifically for this user (what time and what meal did the user eat)

- guestCount: Number of guests for this order
- guestDescription: Some information for the management about the guests
- numberOfMeals: It is a total sum of meals for this order. (guestCount + userOrdered)
- comment and rating: Used for meal feedback of the employee
- mealId and mealName: Additional information needed for orders

Design

When our team started designing the app, our primary goal was to achieve a design that was simple yet elegant. Initially, we opted for a minimalist approach, focusing on a clean layout with blue UI components set against a light background. This design decision was driven by the desire to create an app that was not only visually appealing but also user-friendly, ensuring that the users could navigate through the app with ease. During the development process, however, we noticed that the client wasn't fully satisfied with the initial design and that it didn't quite meet their expectations. This motivated us to revisit our design strategy. We introduced a dynamic linear gradient background, blending the two primary colors of the company, which gave a new life to the app's aesthetic. To maintain clarity and readability, we kept the UI components and text in a white shade. Furthermore, recognizing the growing preference for dark themes among users, we invested considerable effort in preparing a dark theme version of the app. This was not just a mere inversion of colors; we carefully selected shades that would be comfortable for the eyes in low-light conditions while keeping the app's aesthetic integrity intact. Throughout the design process, our focus remained on creating an intuitive user experience. Each component was constructed with the user's journey in mind, ensuring that the app's functionality was as straightforward as it was effective.

Project Management

Team Organization

Team members:

- Sophia Soares Silva – Frontend
- Savo Simeunovic – Backend
- Marin Susic - Backend
- Vasiliy Klyosov – Frontend
- Yatharth Upmanue – Frontend / Scrum master / Project Manager

Process Model: The process model used for this project was AGILE (Scrum)

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

1. Requirements gathering: In this phase, we defined the requirements. We planned the time and effort needed to build the project. Based on this information, we evaluated technical feasibility.

2. Design the requirements: When we identified the project, we worked with stakeholders to define requirements.

3. Construction / iteration: When the team defined the requirements, the work began. Designers and developers started working on the project, which aimed to deploy a working product. Since the product will undergo various stages of improvement, the first task was to finish the basic requirements with simple and minimal functionality and then move on to the optional features and functionality

4. Testing: In this phase, the team examined the product's performance and looked for the bugs.

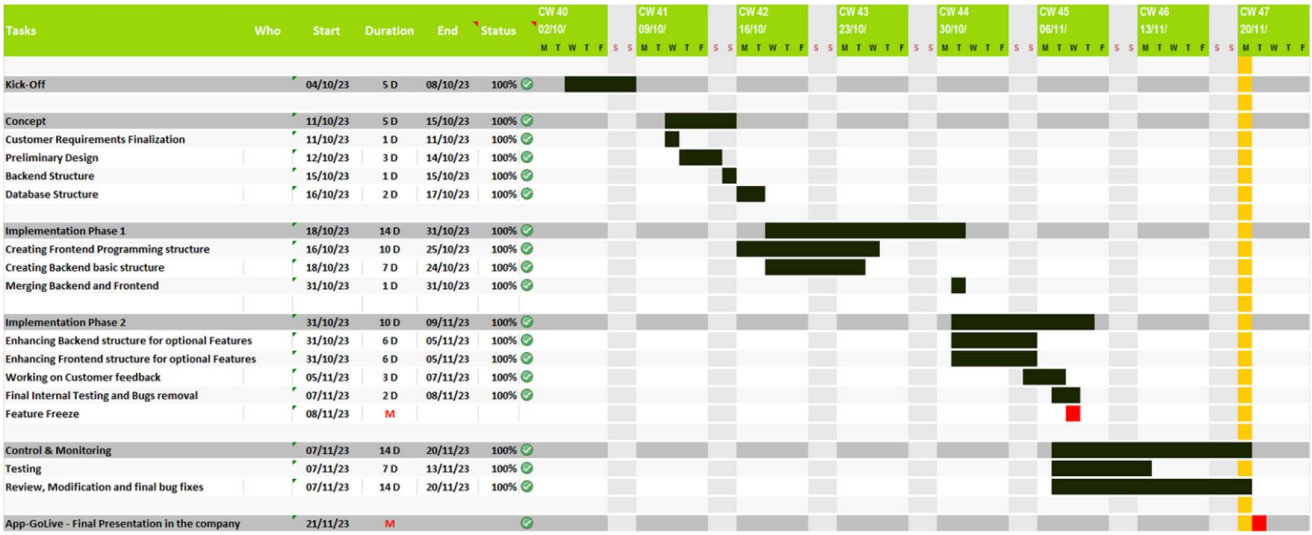
5. Deployment: In this phase, the team issued a product for the user's work environment to test further.

6. Feedback: After releasing the product, the last step was to get feedback from the customer. In this, the team received feedback about the product and works through the feedback. After this feedback session, final changes were made based on the feedback and a full working prototype was released.

Since this was a pretty small project, the managing style adopted by the team was a Democratic style. All the members took part in decision-making, where everything is agreed by the majority. This style was particularly useful when complex decisions needed to be made that required a range of specialist skills. From the overall organizational point of view, job satisfaction and quality of work definitely improved. However, the decision-making process was slowed down due to different working styles of the project members, and the need of a consensus sometimes avoided taking the best decision for the project.

Timeline of the project

Tasks	Who	Start	Duration	End	Status
Kick-Off		04/10/23	5 D	08/10/23	100%
Concept		11/10/23	5 D	15/10/23	100%
Customer Requirements Finalization		11/10/23	1 D	11/10/23	100%
Preliminary Design		12/10/23	3 D	14/10/23	100%
Backend Structure		15/10/23	1 D	15/10/23	100%
Database Structure		16/10/23	2 D	17/10/23	100%
Implementation Phase 1		18/10/23	14 D	31/10/23	100%
Creating Frontend Programming structure		16/10/23	10 D	25/10/23	100%
Creating Backend basic structure		18/10/23	7 D	24/10/23	100%
Merging Backend and Frontend		31/10/23	1 D	31/10/23	100%
Implementation Phase 2		31/10/23	10 D	09/11/23	100%
Enhancing Backend structure for optional Features		31/10/23	6 D	05/11/23	100%
Enhancing Frontend structure for optional Features		31/10/23	6 D	05/11/23	100%
Working on Customer feedback		05/11/23	3 D	07/11/23	100%
Final Internal Testing and Bugs removal		07/11/23	2 D	08/11/23	100%
Feature Freeze		08/11/23	M		
Control & Monitoring		07/11/23	14 D	20/11/23	100%
Testing		07/11/23	7 D	13/11/23	100%
Review, Modification and final bug fixes		07/11/23	14 D	20/11/23	100%
App-GoLive - Final Presentation in the company		21/11/23	M		



The timeline for the project was established prior to its commencement and was consistently maintained and updated throughout. The original schedule for the project, including the dates and durations, remained unchanged, and the project progressed according to the planned timeline.

The project was divided into four major phases, as outlined in the timeline. The first phase, the conceptual phase, involved the Project Manager meeting with the customer to comprehend their needs and requirements. Subsequently, the Project Manager conveyed this information to the team, who then initiated the preliminary design and basic backend structure of the app. Following this, the updated design was presented to the customer for approval to proceed with the programming phase, which lasted for 5 days.

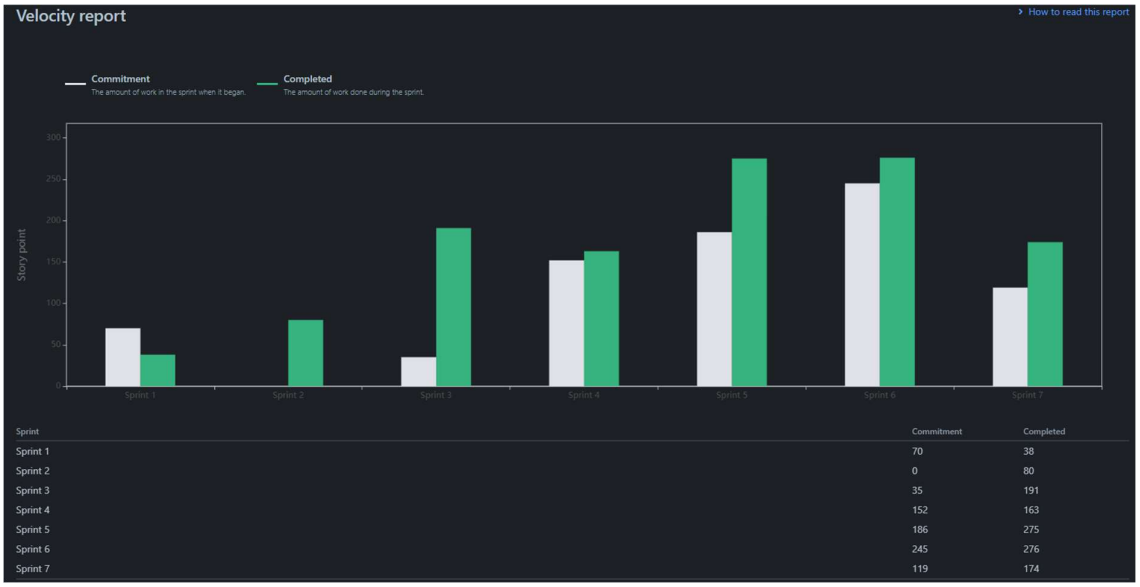
The first implementation phase followed, during which the team collectively began programming the app, focusing primarily on the essential features required by the company. This phase, which emphasized basic functionality, was successfully completed within 14 days, culminating in a review with the customer.

The subsequent 10 days were allocated to the second implementation phase, during which the team concentrated on incorporating optional features such as the food buzzer and wish list. This phase was characterized by the team's enthusiasm for enhancing the app with new features. A feature freeze was then implemented, signifying a shift from introducing new features to refining and perfecting the existing ones.

The Control and Monitoring phase, spanning 14 days, followed. The initial 7 days were dedicated to testing, during which the customer received the first official prototype of the app for evaluation and feedback on their user experience. Subsequently, the team progressed to the final 7 days of the project timeline, focusing on bug fixing, code refactoring, and implementing a new user interface to enhance the app's usability.

As planned, the app was launched with a presentation at the company, attended by over 50 employees, including the Head of the Department. The project was deemed a resounding success, as the team effectively navigated numerous challenges and surpassed the customer's expectations.

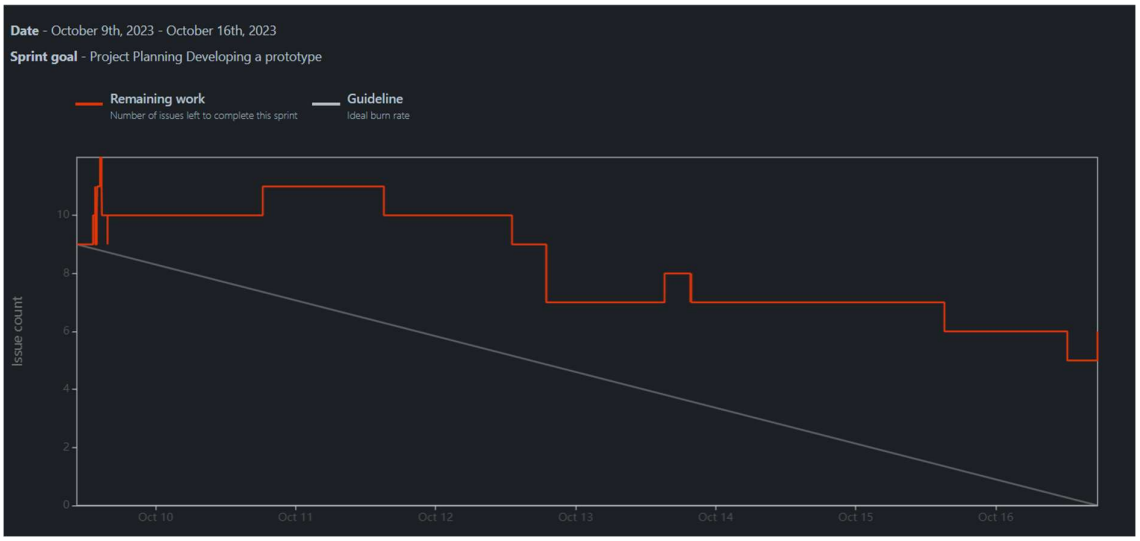
Velocity Report



Sprint	Commitment	Completed
Sprint 1	70	38
Sprint 2	0	80
Sprint 3	35	191
Sprint 4	152	163
Sprint 5	186	275
Sprint 6	245	276
Sprint 7	119	174

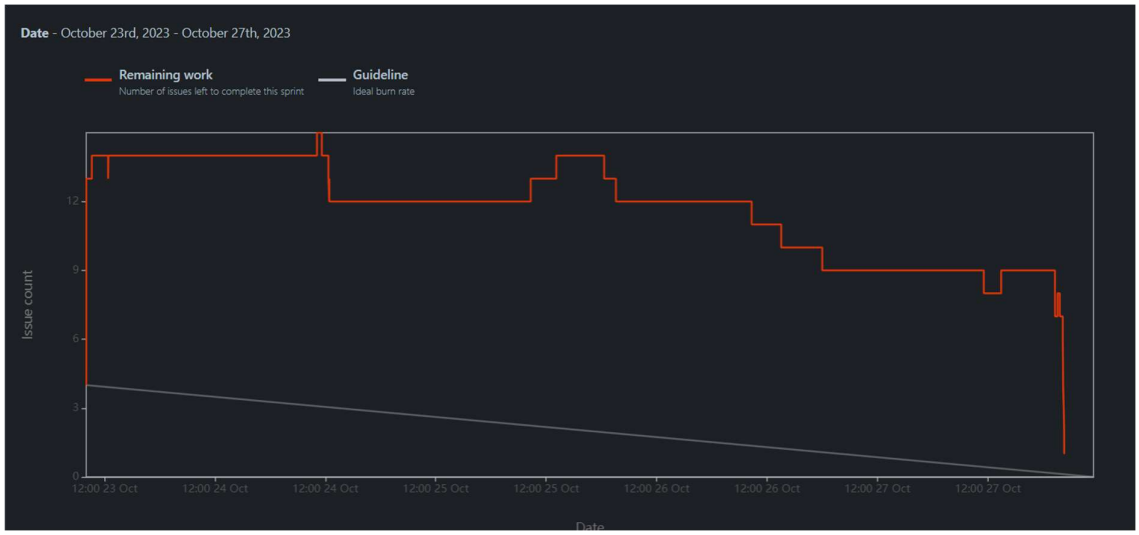
Sprints

Sprint 1:



Sprint 2:

Sprint 3:



Sprint 4:

Sprint 5:

Sprint 6:

Sprint 7:

In the initial stages of the project, our approach to managing sprints faced challenges, resulting in suboptimal outcomes. However, as the project progressed, we made significant improvements in our sprint planning and execution, leading to more effective and successful sprints towards the project's conclusion.

Early sprints were characterized by certain shortcomings, such as unclear sprint goals, inadequate task breakdown, and insufficient consideration of team capacity and dependencies. As a result, the team encountered difficulties in meeting sprint objectives, experienced delays, and faced challenges in delivering the intended scope of work within the sprint timeframe.

To address these issues, we implemented several improvements in our sprint management approach:

1. **Enhanced Sprint Planning:** We refined our sprint planning process by establishing clear sprint goals, breaking down tasks into smaller, manageable units, and ensuring that the team had a comprehensive understanding of the sprint backlog. This improved clarity and alignment within the team, enabling a more focused and purposeful approach to sprint execution.

2. **Capacity and Dependency Management:** We placed greater emphasis on understanding team capacity and identifying interdependencies between tasks. By considering team members' availability, skill sets, and potential task dependencies, we were able to allocate work more effectively and anticipate potential bottlenecks, leading to smoother sprint execution.

3. **Iterative Improvement:** Throughout the project, we embraced an iterative approach to process improvement. We actively solicited feedback from team members at the end of each sprint, identifying areas for enhancement and implementing adjustments to our sprint planning and execution processes. This iterative feedback loop allowed us to adapt and refine our approach based on real-time insights and experiences.

As a result of these improvements, we observed a noticeable shift in the quality and effectiveness of our sprints. Towards the middle and latter stages of the project, our sprints became more structured, focused, and productive. The team demonstrated improved cohesion, a clearer understanding of sprint objectives, and a more refined approach to task execution and delivery.

By leveraging these insights and refining our sprint management practices, we were able to achieve better alignment, productivity, and outcomes in our sprints, ultimately contributing to the successful completion of the project.

Member Contribution

Sophia Soares Silva

As a key member of the team, my contributions significantly shaped both the technical and collaborative aspects of our project. My primary role as the lead front-end developer involved the creation and implementation of several user interface components, ensuring both functionality and aesthetic appeal. My responsibilities included building most of the app's views, which involved a detailed process of designing, coding, and testing to guarantee an intuitive and user-friendly experience. I took the initiative to regularly refactor the front-end codebase, a critical task that maintained the code's efficiency and readability, especially as the complexity of the project evolved. Collaboration and knowledge sharing were key elements of my role. I frequently engaged in reviewing and refactoring the front-end code written by my colleagues. I was also proactive in assisting my teammates with front-end-related challenges, promoting a team environment conducive to learning and mutual growth.

Beyond coding, I played an important role in the app's design process alongside another team member. We worked together to conceptualize and finalize the app's design elements, focusing on creating an engaging and accessible user interface. Furthermore, I consistently participated in all team and professor meetings.

In summary, my involvement in this project was comprehensive and varied. I took pride in being a driving force in the front-end development, a collaborative team player in design and code refinement, and an active participant in the execution of the project. My contributions not only advanced the technical aspects of the app but also reinforced a culture of teamwork, innovation, and collective problem-solving.

Savo Simeunovic

- Design and implementation of the noSQL Firestore database
- Design and implementation of all of the viewmodels and their relationships between each other.
- Creating most of the Firestore service CRUD functions.
- Localization of every hard-coded string in the project.
- Providing function to retrieve AI generated images and translate the prompt from German to English using translation API

- Significant design improvements implementation of the Custom Calendar widget
- Statistics view: number of orders per day, design and implementation (both frontend and backend)
- Custom allergies design and implementation (both frontend and backend)
- All unit tests for both viewmodels and database (together with database mocking)
- Contributing to bugfixing in both frontend and backend

Vasiliy Klyosov

Design and UI Development

- Identified and structured the fundamental design layout for the application, while initiated the Figma project, creating the basic UI design for the entire view, from which together with the team we expanded further.
- Incorporated the company's design guidelines and collaborated closely with the customer to align design decisions.
- Designed the primary views for Employees and Chefs, notably the calendar view for meal selection, as well as developed essential widgets for within the design view.

Development:

- Made significant contributions to frontend development by crafting views and providers for various screens and frames within the Employee and Chef views.
- Conducted a comprehensive refactor of the main view to incorporate methods for improving local data persistence without data loss while handling data manipulation.
- Assisted in Backend team by modifying the functions, manipulating data retrieval functions.

Collaboration and Coordination:

- Actively engaged with the Backend team, ensuring seamless integration between frontend and backend functionalities, ensuring backend functions retrieved specific attributes from the database accurately.
- Participated in team meetings, contributed to collective decisions, and ensured development alignment with customer requirements.

- Played a role in documentation creation, focusing on both code documentation and project documentation, refining and enhancing both aspects for clarity and completeness.

Marin Susic

- Implementation of Authentication services (login, register, confirm email, verify email confirm, etc.)
- Language drawer and localizations (translations)
- Cloud functions for notifications both email and in phone
- Notifications (backend) - on message handlers, types of notifications, automated notifications
- Add guests to an order (implementation)
- Edit UI so it is grayed out after time to order has expired
- Manage the CI/CD Pipeline
- Creating the firebase connection of the project with firebase (this was a big bug, since originally this was not possible)
- Adding firebase security rules
- Creating util functions for easier and reusable code (conversion, checks, etc.)
- Wishlist feature complete implementation
- Commenting the backend code for better (dart doc) documentation
- Dark mode toggle implementation
- Implementing functions in the backend for communication with the backend (i.e. checkIsChef(), retrieveUserOrders(String id), etc.).
- Bug fixing (such as: Menu staying created after all meals are deleted, getting multiple snack bars after ordering for multiple people “order sent”, email notification sent only if menu exists, not always, etc.)

Yatharth Upmanue

As the Project Manager and Scrum Master, I played a pivotal role in shaping the project and ensuring its successful execution. I was responsible to define the project's objectives, create a detailed work plan, allocate resources, and monitor progress to ensure the project is completed on time and within budget. Additionally, my role was to manage risks,

communicate with stakeholders, and lead the project team to achieve its goals. As a Scrum Master I played a crucial role in facilitating the successful implementation of the Scrum framework within the team. I guided the team through various Scrum events such as Sprint Planning, Daily Stand-ups, Sprint Reviews, and Sprint Retrospectives. I tried to foster an environment where the team can effectively collaborate and deliver value. Removing obstacles that hinder the team's progress, protecting the team from distractions, and facilitating productive meetings and collaboration were also integral to my responsibilities. Furthermore, I was responsible for fostering a culture of continuous improvement within the team and advocating for Agile principles.

In summary, my role as a project manager and scrum master encompassed a wide range of activities aimed at ensuring that projects are completed successfully and on time. This includes project planning, resource allocation, risk management, stakeholder communication, team leadership, project monitoring and control, quality assurance, and project closure. My overarching goal was to ensure the successful delivery of project while maintaining high standards of quality, timeliness, and stakeholder satisfaction.

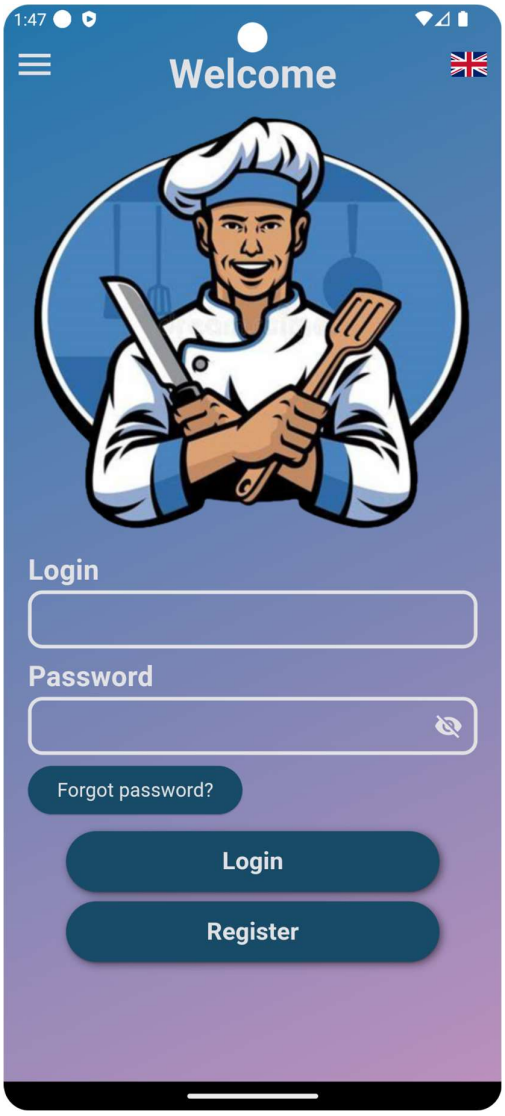
In addition to my managerial role, I collaborated with a team member to conceptualize and finalize the app's design and basic structure for the customer. I actively assisted my colleague in reviewing and refactoring frontend code and addressing the frontend related challenges. Moreover, I undertook multiple tasks in the frontend such as creating the basic calendar function, creating notifications, implementing dark mode, commenting the frontend code for documentation purposes and helping out with more such tasks. I also created comprehensive project documentation to ensure clarity for all stakeholders, presentation for the University and the company and designed the poster together with a colleague. Lastly, I also was independently responsible to make the app work in IOS and despite many challenges, I was successfully able to do so.

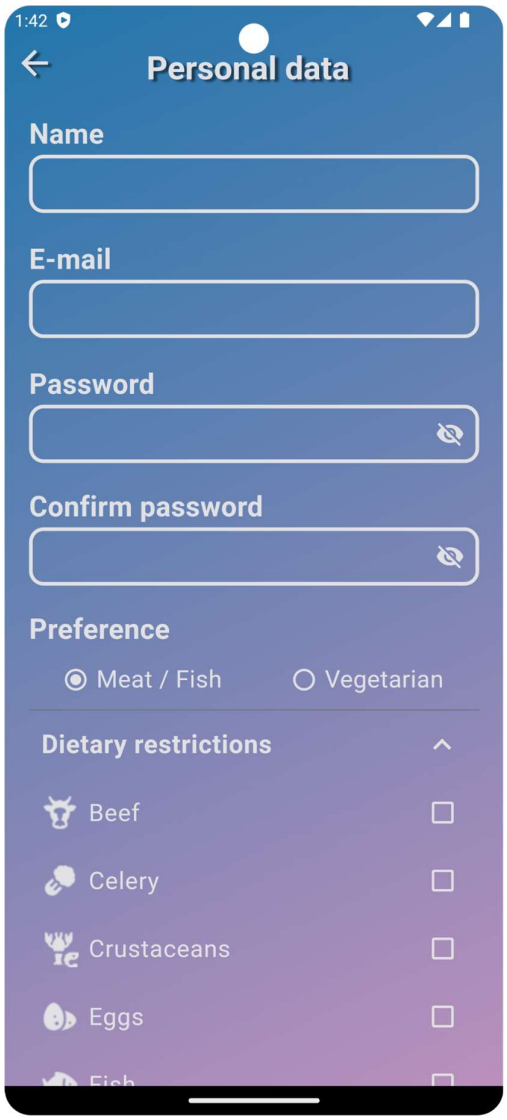
My involvement in this project was all-rounded, encompassing project planning, team collaboration, design, development, and documentation, reflecting my commitment to ensuring the project's success from a project manager's perspective as well as a developer perspective.

Result

UI TOUR

Login and register screens

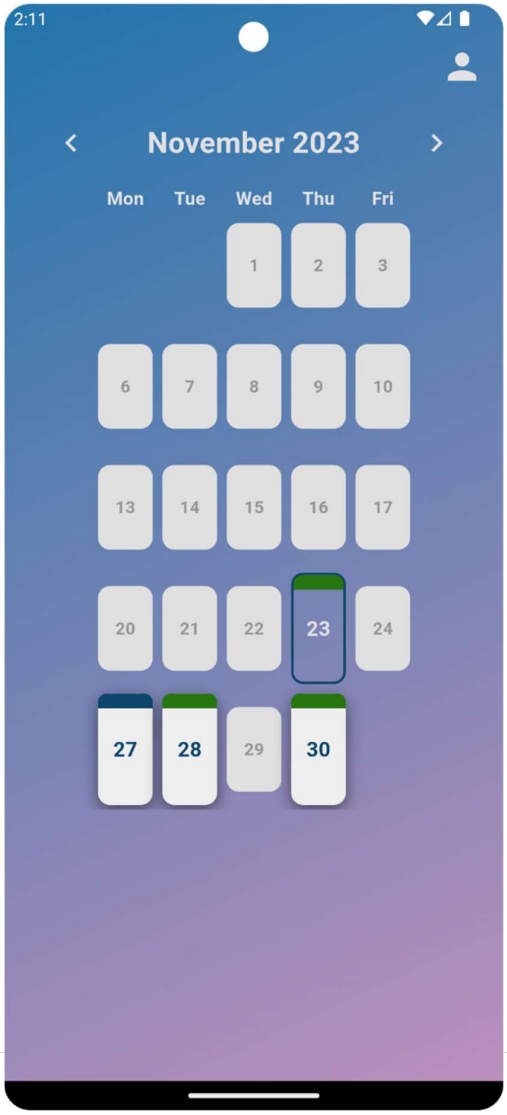


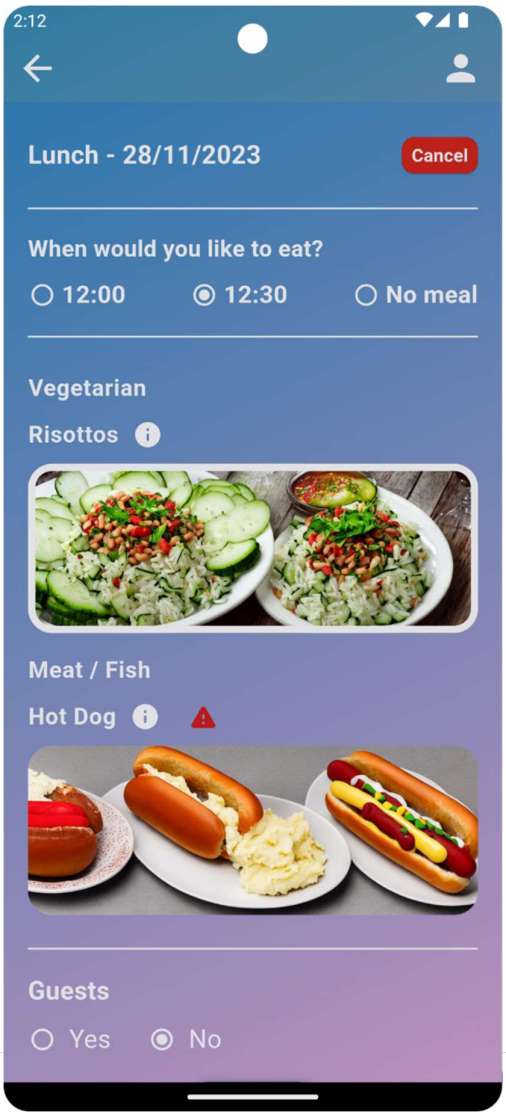


- After opening the kitchen module, the user is brought to the login screen of the module.
- The user can enter their credentials and log into the app (if they already have an account).

- If the user does not have an account, they can click on the register button, taking them to the registration screen.
- The registration screen contains a form that the user needs to fill in.
- Key parts of the register form are the email and password fields.
- Users can register only with an “exxcellent.de” email account to prevent outsiders from ordering meals and causing potential damage.
- The register screen also contains a list of known allergies for the user to pick from, which is later used to warn the users of meals containing ingredients they are allergic to.
- After filling out the form, the user can click confirm and will be taken to a screen asking them to confirm their email.
- The user will not be able to proceed until the email is confirmed.

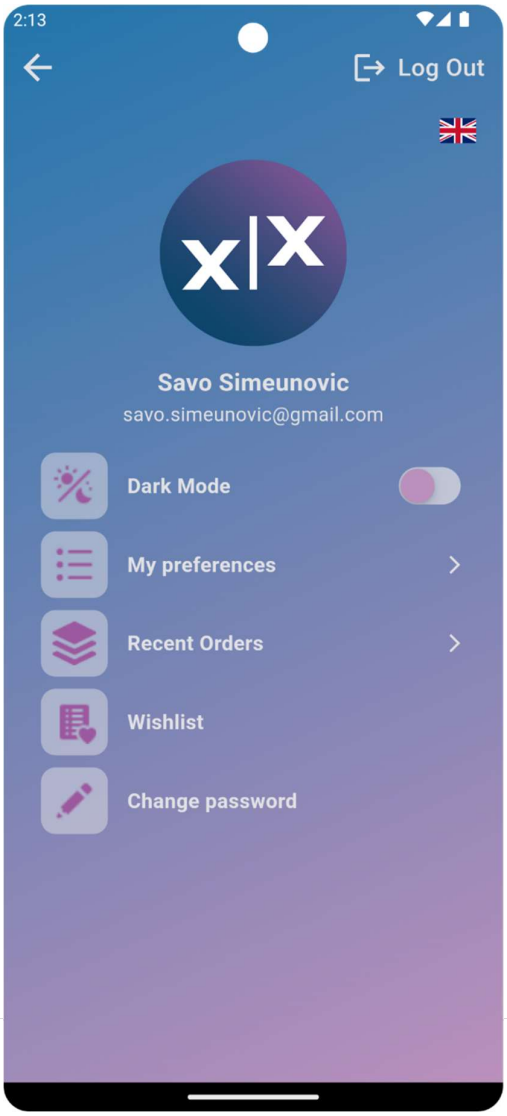
The calendar view

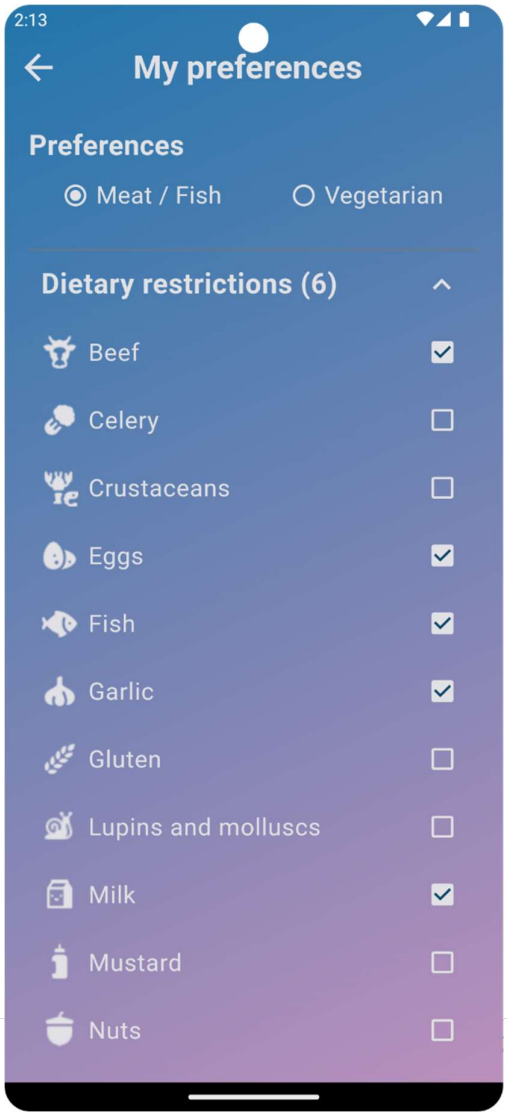




- After the user logs in, they are taken to the calendar view.
- Past days are unclickable, while future days with a menu available are shown with a blue top on top of the date.
- Future days with a menu and the user's previous orders are shown with a green top.
- On the right, the user can see how it looks when creating an order for a menu.
- Meals are displayed along with available time slots, and the user may place an order.
- If the user has guests, they can select "yes" in the “Guests” radio button.
- This action will open a new field allowing the user to declare which meals their guests would like to have, at what time, and who the guests are.

The profile view

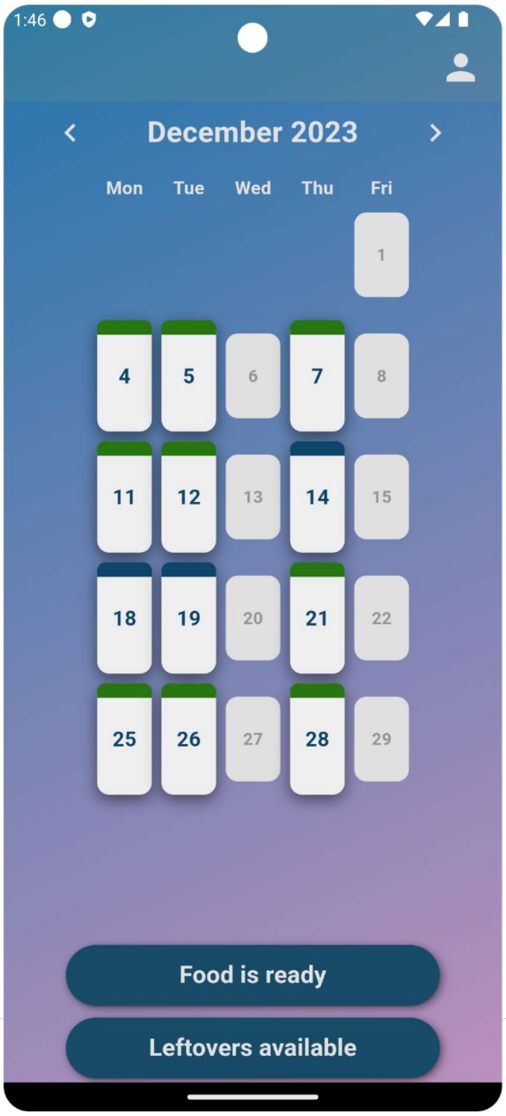


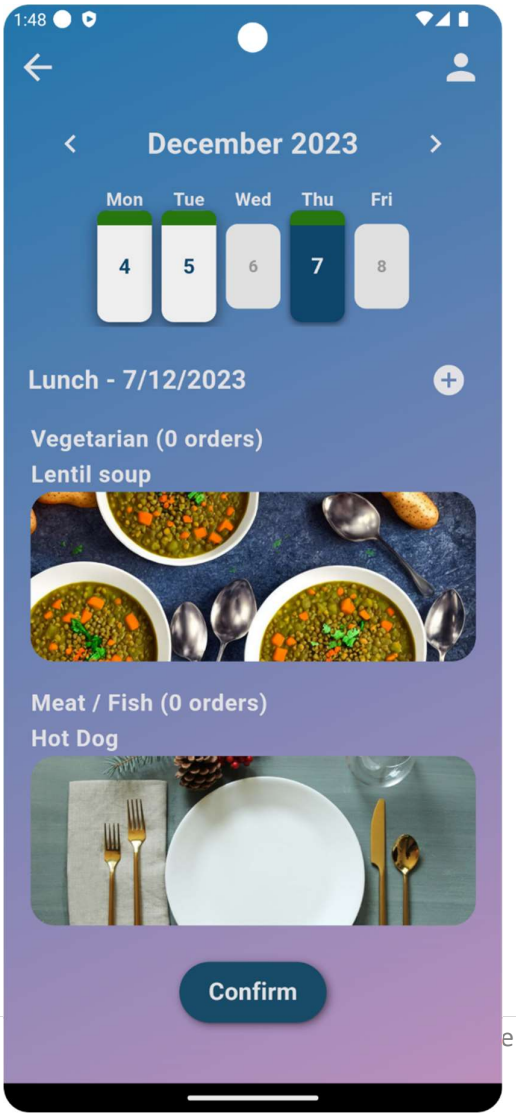


When the user clicks on the person icon in the top right they are taken to their profile screen, this screen allows them to:

- Change the app language
- Edit their preferences (vegetarian, non-vegetarian, allergies, and custom allergies submission)
- View and rate their recent orders
- Add items to their wishlist
- Submit a change password request

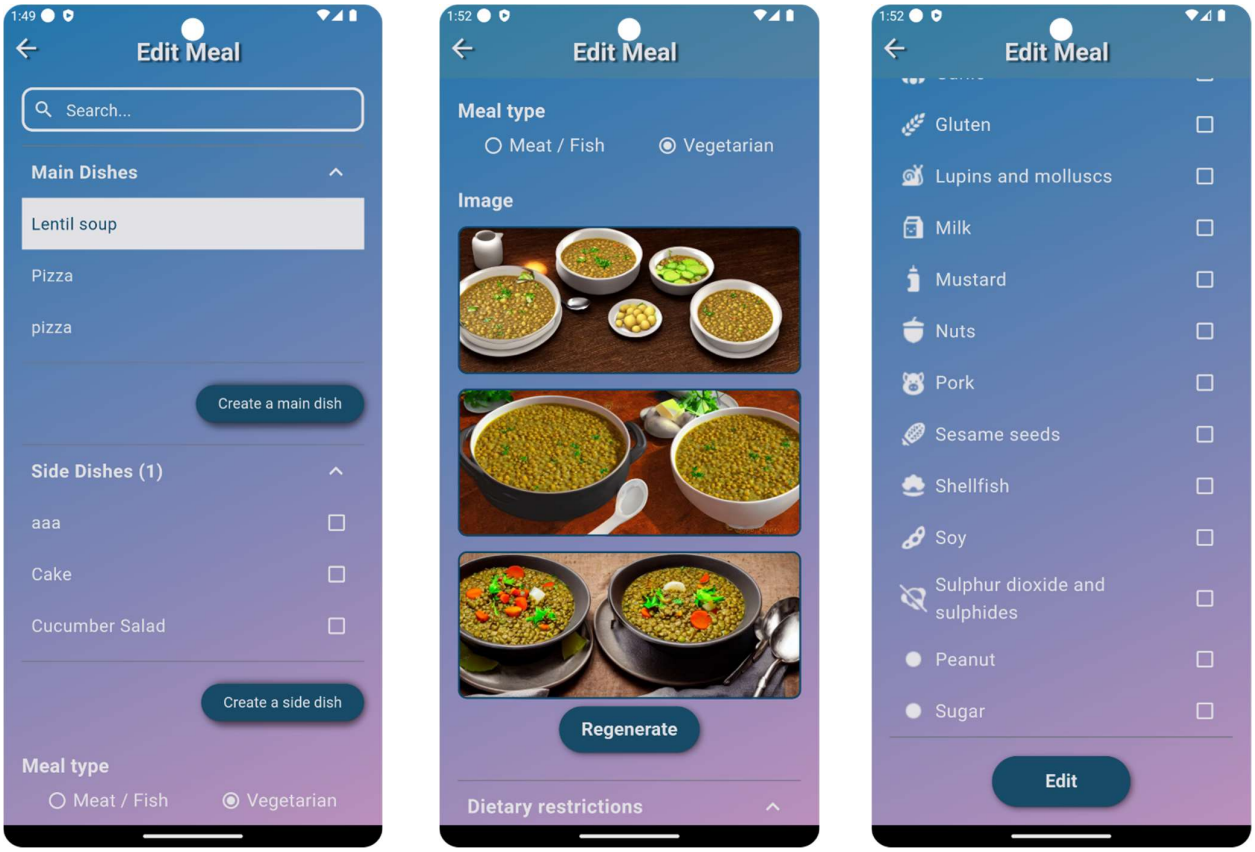
The chef calendar and create meal views





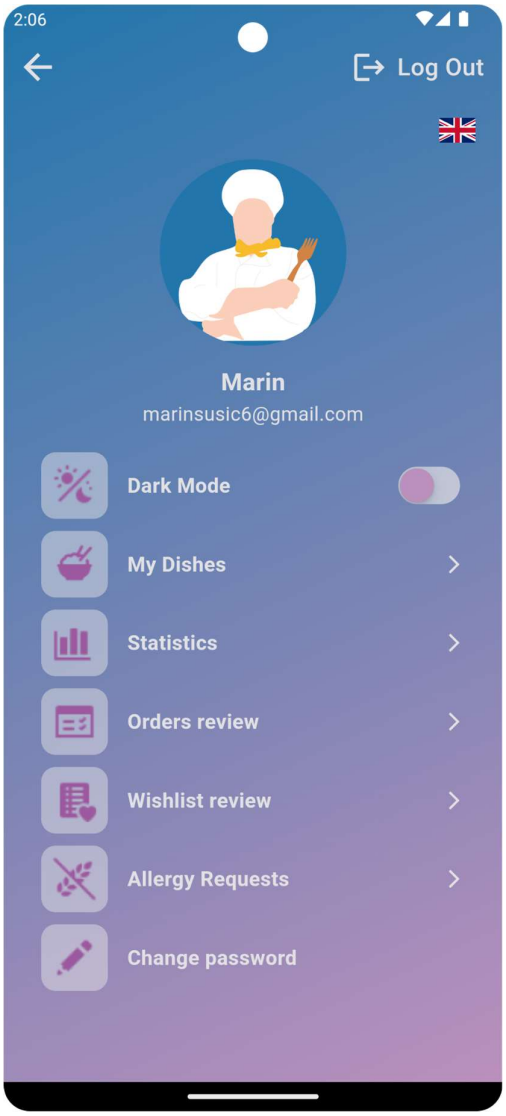
- After logging in, the chef is taken to the calendar view, where they can see all the working days for creating a menu.
- Dates with a blue or green tick on top indicate availability, with blue meaning the date is available but no menu has been created, and green meaning the date is available with an existing menu.
- The chef can send notifications to the users from this view.
- The "food is ready" notification sends a notification to all users who have ordered for the current date, while the "leftovers available" notification informs users about available leftovers, reducing food waste and allowing those with small portions to have more.
- Clicking on a date takes the chef to the meal edit or meal create view (on the left).
- Here, the chef can select a meal to edit and will be taken to the meal editing screen.
- The chef can also delete meals/menus, notifying users that the menu has changed.

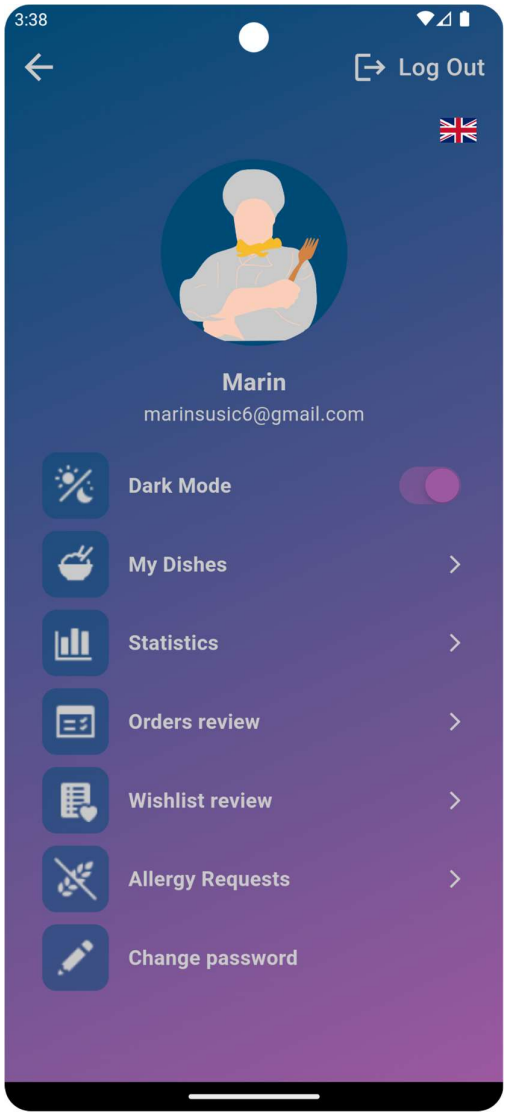
Chef edit meal view



- After clicking on a given meal inside the edit menu screen, the chef is taken to the edit meal screen.
- Here, the chef can change/create main dishes, change/create side dishes, and modify allergic ingredients in the meal.
- The chef also has the ability to choose an image for the meal.
- If the chef decides not to pick an image, the meal will only have a placeholder as an image.
- If the chef selects a picture, they can click "generate," and the main dish's name, along with the names of the side dishes, will be sent to an AI API to create 3 images for the chef to choose from.
- If the chef doesn't like any of the images generated, they can press "regenerate" to create 3 more images, repeating this process until they find the suitable image for their meal.
- After creating/editing the meal, the chef can click the "confirm" button at the bottom of the screen, and the meal will be created and temporarily stored in the menu view.
- Once all changes are made to the menu, the chef can click "confirm" again to push this data into the database, completing the menu creation and allowing users to place orders.

Chef profile screen

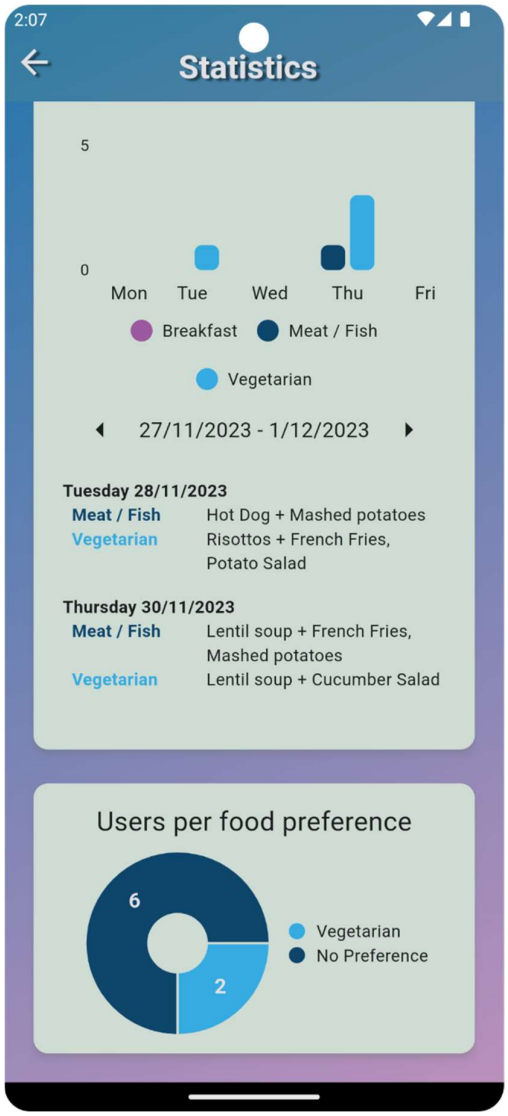


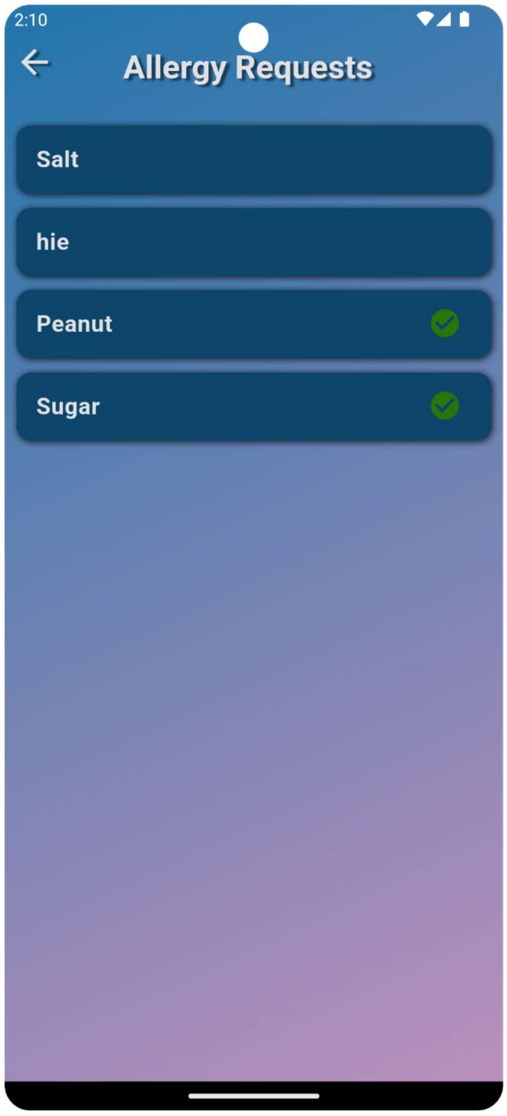


- From the calendar view, the chef can access their profile screen by clicking on the person icon in the top right corner of their screen.
- In the profile screen, they can change their application mode from light to dark mode and vice versa.
- They can also view and edit their dishes by clicking on a meal and then clicking an edit button.
- Additionally, they can check statistics, review their orders (including user ratings and comments), view user wishes (and delete them once fulfilled or if the chef doesn't approve), and review custom allergy submissions.

- Furthermore, they have the option to change their password, which will trigger an email requesting a password change.

Chef statistics and custom allergies

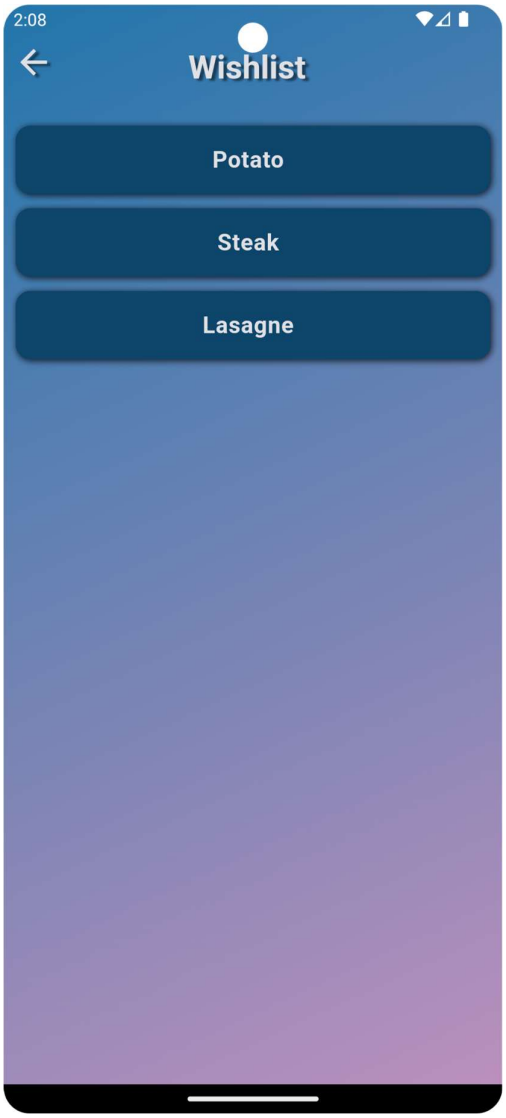


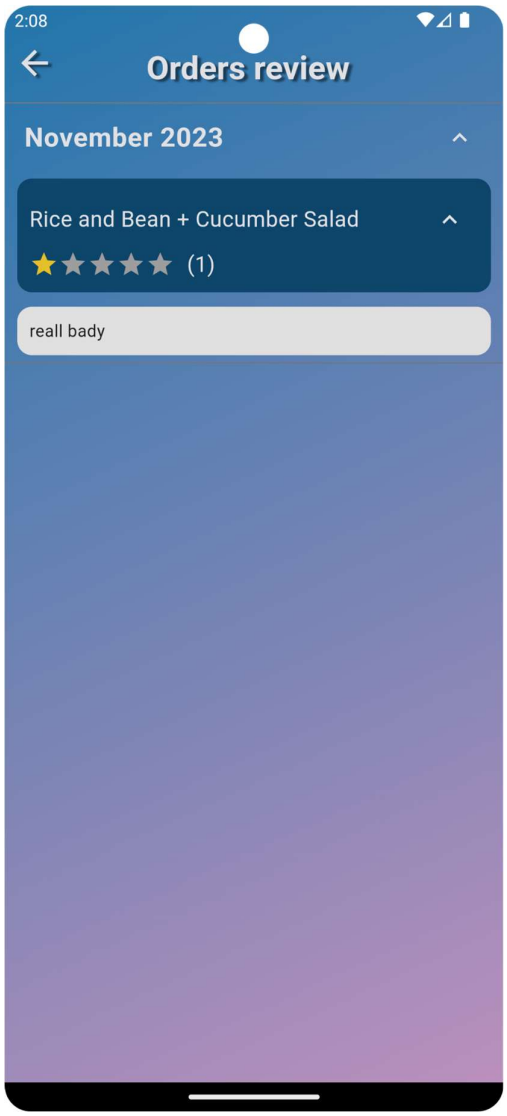


- By clicking on the statistics from the profile view, the chef can see the statistics of how many meals have been ordered of each type on each day of a given week.
- The chef can also view a breakdown of the meals they have cooked that week, providing insight into which meals received the corresponding amount of orders shown in the chart.

- Additionally, there is a pie chart displaying the number of registered users who are vegetarian and those who are not, enabling the chef to make assumptions or predictions about future order quantities.
- On the right, the custom allergies screen can be accessed. Here, the chef can either accept or deny the allergies that users have submitted.
- If an allergy is accepted (chef swipes from left to right), it will be added to the confirmed allergies list, marked with a checkmark, and made available to all users.
- If the allergy is denied (chef swipes from right to left), it will be deleted, and no one will be able to see it.

Chef wishes and review overview





- The first image shows the Wishlist screen, where the chef can view the wishes submitted by users. Once a wish has been fulfilled or if the chef does not wish to fulfill it, they can remove it by swiping from right to left on the wish.
- On the right, the reviews screen displays how users have rated the given meals. Meals are organized by months, allowing the chef to easily locate a specific meal. After selecting a meal, the chef can view the average rating in stars and the number of ratings received in brackets. Users can also leave comments on ordered meals, and the chef can access an overview of these comments by clicking on the arrow at the far right of the order container.

Languages support

Considering that the users may be coming from all around the world, the application also supports localization in 15 languages and has been programmed so that adding a new language is a matter of providing the flag of the country and translating the strings inside of a file. The app supports the following languages:

→ Bulgarian
→ Albanian
→ Croatian
→ German
→ Serbian
→ Turkish
→ Russian
→ Portuguese (Brazil)
→ French
→ Danish
→ Swedish
→ Italian
→ English
→ Spanish
→ Hindi

Achievements and Limitations

Our team achieved not only the completion of the required features but also the successful implementation of all optional features and the addition of extra functionalities to the app, including Dark mode, multiple language support, and custom allergies.

However, we encountered several challenges during the development process:

- Working with previously created code: Initially, we faced difficulties working with existing code that was not originally developed by our team.
- Working with deprecated software: Compatibility issues arose due to the use of outdated software by the previous developers, while our team utilized up-to-date software versions, leading to various complications.
- Connecting issues between the previous and the new code: We encountered challenges when attempting to integrate our project with the existing codebase, which resulted in unforeseen difficulties.
- Establishing and connecting Firebase with the previous code: Integrating Firebase with the existing codebase posed challenges, requiring significant effort to ensure seamless connectivity.
- IOS: Overcoming obstacles to ensure the smooth operation of the iOS application following minor code modifications.

Each of these challenges required careful consideration and problem-solving to ensure the successful completion of the project.

Customer Feedback

- "The students **started the project professionally** and with commitment from the very first minute."

- "I was particularly **impressed** by the **students' independent way of working**. We only had to act as Customers"

- "Their **project plan was well thought out and tailored to the customer's satisfaction**
    - a functional prototype after 4 weeks for testing,
    - a professional presentation of the prototype to the customer including known features
    - **actively obtaining feedback** and enough time for bug-fixing
    - **implementing customer's feedback**)."

- "The students **implemented far more functionalities** than expected, delivered **good code quality**, thought about the **appropriate architecture** and wrote **quality-assuring tests**."

Lessons Learned

The lessons learned from the project are valuable insights that can significantly impact future projects. Let's delve into each of these lessons in detail:

1. **Unexpected issues can occur at any time:** Throughout the project, we encountered unexpected challenges, such as working with previously created code, compatibility issues with deprecated software, and difficulties in integrating Firebase with the existing codebase. These unforeseen issues required adaptability, problem-solving skills, and a proactive approach to address them effectively. It's essential to anticipate and prepare for unexpected hurdles in future projects, fostering a mindset of flexibility and resilience within the team.

2. **Teamwork requires patience due to different working styles and work ethics:** Our experience highlighted the importance of patience and understanding in a team setting. Each team member possesses unique working styles, perspectives, and approaches to problem-solving. Embracing these differences and fostering an environment of mutual respect and collaboration is crucial for achieving cohesive teamwork. Effective leadership and clear communication can help align diverse working styles towards a common goal, promoting a harmonious and productive team dynamic.

3. **Importance of regular resting:** The project emphasized the significance of regular resting and maintaining a healthy work-life balance. The demanding nature of the project underscored the need for adequate rest and recuperation to prevent burnout and maintain overall well-being. Encouraging team members to prioritize self-care, take breaks when needed, and manage workload effectively can contribute to sustained productivity and prevent exhaustion.

4. **Effective communication is the key:** Clear and open communication played a pivotal role in overcoming challenges and ensuring project success. From addressing unexpected issues to coordinating tasks and sharing progress updates, effective communication fostered transparency, alignment, and a shared understanding among team members. Encouraging open dialogue, active listening, and constructive

feedback can enhance collaboration, mitigate misunderstandings, and promote a cohesive team environment.

By reflecting on these lessons learned and integrating them into future projects, our team can leverage these insights to enhance project management, teamwork, and overall project outcomes.

Outlook and Further developments

Feature Roadmap

Feature 1: Redirecting the Chef from Statistics to Order Review

Current Functionality: At present, the Chef interface primarily allows viewing statistics related to meals created. This includes a candle chart depicting the number of people who have ordered, along with the distribution of vegetarian and non-vegetarian meal orders. Additionally, there's an optional pie chart showcasing the classification of employees into vegetarian and non-vegetarian categories.

Proposed Modification: The modification involves enhancing the Chef interface to enable a smooth transition from the statistics view to an Order Review section. Upon implementing this feature, chefs will have direct access to a consolidated summary of orders.

Implementation Details:

User Interface (UI): Introduce a button or an intuitive UI element within the statistics view that leads directly to the Order Review section. Design the transition to maintain a consistent visual style and user experience.

Backend and Data Handling: Modify backend logic to ensure seamless retrieval and display of order-related data when transitioning from statistics to the Order Review section. Update data retrieval mechanisms to populate the Order Review section with relevant information, such as order statuses, details, and order history.

Navigation and App Flow: Reconfigure app navigation to ensure smooth transitions between the statistics and Order Review sections. Ensure that chefs can easily navigate back to the statistics if needed.

Adaptability within the Codebase: Adapting the codebase for this feature primarily involves UI modifications, navigation restructuring, and backend adjustments. The existing data models and database structure might require minor alterations to accommodate additional order-related information.

Feature 2: Redirecting the Chef from Wish List to Meal Creation

Current Functionality: The WishListView currently displays the employee-generated wish list items for the chef's review. The WishProvider class retrieves and manages data related to wishes from the database.

Proposed Modifications: Enable chefs to convert selected wishes from the list into meals available for ordering within the app.

User Interface (UI): Modify the WishListView UI to allow chefs to interact with individual wish list items to create meals. Add functionality to trigger meal creation based on selected wish list items.

```
void _createMealFromWish(BuildContext context, String wish, WishProvider
provider) {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      // Dialog setup for meal creation
      actions: [
        TextButton(
          onPressed: () {
            provider.createMealFromWish(wish);
            Navigator.pop(context); // Close the dialog after meal
creation
          },
          child: Text('Create Meal'), // Customize button text
        ),
      ],
    ),
  );
}
```

Backend and Data Handling: Update the WishProvider class to include a method facilitating the creation of meals from selected wishes in the database.

```
class WishProvider with ChangeNotifier {
  final FirestoreService firestoreService = FirestoreService();

  void createMealFromWish(String wish) {
    firestoreService.createMealFromWish(wish);
    notifyListeners();
  }
}
```

Navigation and App Flow: Consider integrating the meal creation functionality into the existing app flow. Ensure that the transition from wish list to meal creation maintains a smooth user experience.

Adaptability within the Codebase: Adapt the codebase to include the new method for creating meals based on wishes within the WishProvider. Integrate the updated WishListView UI to enable chefs to create meals from selected wishes.

Feature 3: Chef Can Add Surprise Dishes for Employees

Objective: The Chef will have the capability to add surprise dishes, which will be reflected on the employee's calendar widget.

Proposed Implementation:

Chef View:

New View for Chef: Create a new view within the Chef's interface to add surprise dishes.

Interface for Adding Surprise Dishes: Design an intuitive interface for the Chef to input details of the surprise dish such as name, description, date, and meal type. Backend Integration: Implement backend functionality to store and manage the surprise dish data.

Employee Calendar Widget:

Marking Specific Day: Modify the employee's calendar widget to display a distinct mark or indicator on days when a surprise dish is added by the Chef.

Integration with Calendar Widget: Ensure the surprise dish information is properly integrated into the calendar widget, allowing employees to view details on selected days.

Technical Implementation:

UI Updates: Implement UI modifications in the Chef view and the employee's calendar widget to accommodate the addition of surprise dishes.

Backend Handling: Set up backend logic to manage the storage, retrieval, and display of surprise dish data.

Database Modifications: Adjust the database schema if necessary to incorporate surprise dish-related information.

Proposed Technical Approach:

Chef's View: Develop a dedicated UI for Chef to input surprise dish details. Implement backend API endpoints to handle data submission and storage.

Employee Calendar Widget: Modify the calendar widget to display a visual marker for days with surprise dishes. Enable the widget to fetch and display surprise dish details when a marked day is selected.

Code Modifications: Integrate UI changes for Chef view using Flutter widgets. Update existing backend services or create new ones to handle surprise dish data.

Database Adjustments: Modify database structure to accommodate storing surprise dish information.

Feature 4: AI-Generated Recipes for Chef Meals and Employee Viewing

Objective:

This feature involves the use of AI to generate recipes for meals added by the Chef. The Chef selects from three AI-generated recipes, and employees can view the recipe associated with their ordered meal on the order details page.

Proposed Implementation:

Chef's Perspective:

AI Recipe Generation: Integrate three different AI systems using three separate API calls to generate recipes for meals added by the Chef. Display the generated recipes for the Chef's review and selection. Provide a user-friendly interface for the Chef to select and associate the preferred recipe with the meal.

Employee's Perspective (Order Details Page):

Viewing Recipes: Display the associated recipe for the ordered meal on the order details page for employees. Allow employees to access the recipe associated with their ordered meal. Enable viewing of the recipe along with order details such as meal name, ratings, and comments.

Technical Implementation:

Chef's View: Develop UI components to facilitate the display and selection of AI-generated recipes. Implement API integration to fetch recipes from three different AI sources. Design UI elements for Chef interaction to associate a selected recipe with a specific meal.

Employee's Order Details Page: Modify the order details page to incorporate a section displaying the associated recipe for the ordered meal. Implement retrieval and display of the recipe information along with other order details. Ensure a seamless user experience for viewing recipes within the order details section.

Code Modifications: Integrate AI service APIs and manage API calls for recipe generation in the Chef's view. Update order details widget to fetch and display associated recipes for employee viewing.

Backend Integration: Manage backend services to store and link generated recipes with specific meals for both Chef and employee access.

Technology Upgrades

Backend Services:

Server-Side Technologies: Consider using server-side languages like Node.js, Python (Django or Flask), or Go to enhance backend services, improve scalability, and handle complex server logic efficiently.

GraphQL: Evaluate migrating from RESTful APIs to GraphQL for more flexible and efficient data querying, reducing over-fetching or under-fetching of data.

Database and Storage:

Relational Databases: Assess if migrating from Firebase Firestore to a relational database like PostgreSQL or MySQL better suits the data structure and querying needs, offering more robust data management and querying capabilities.

Object Storage: Consider utilizing cloud-based object storage like Amazon S3 or Google Cloud Storage for efficient storage and retrieval of multimedia content.

Microservices Architecture:

Containerization and Orchestration: Adopt containerization with Docker and orchestration with Kubernetes to build a microservices architecture, enabling easier scaling, maintenance, and deployment of individual services.

Code Refactoring and Modernization:

Utilize Modern Language Features: Upgrade the codebase to leverage the latest features and functionalities of Dart and Flutter, improving code readability, maintainability, and performance.

Continuous Integration/Continuous Deployment (CI/CD):

Automation Tools: Implement CI/CD pipelines with automation tools like Jenkins, GitLab CI, or GitHub Actions to streamline development, testing, and deployment processes.

Performance enhancements

- Some of the data retrieval functions take a few seconds to complete (such as getting all employee orders). There could be a better way to optimize it.
- In multiple places, there are unnecessary calls to the database which can be omitted (just perform operations locally).
- Asynchronous database calls: currently, every call awaits for the database response which significantly decreases the application performance. Some of them could be done asynchronously but this has to be done with care.

Quality assurance

- Currently, there are some unit tests for the functions of viewmodels that test if functions do what they are intended for. They could be improved with some edge cases.
- Authentication service is not being unit tested
- There are no integration or acceptance testing done.
- Widget tests and important views tests.

Administrator Implementation

- Implement an admin user that can change the Chef's working days.
- Give Chef role to users without touching the database (from UI)
- Remove some users from application (if user leaves the company)

Design and Usability Enhancements

In terms of design and usability, some enhancements could be made based on insights from user tests and UX research. A possible approach could be to focus on improving the user interface to ensure a more intuitive and engaging experience for users with different needs. This includes implementing adaptive features for those with vision impairments, such as high-contrast color schemes and larger text options, ensuring the app is easily navigable and readable. Another option could be to integrate advanced voice control capabilities, allowing users to speak commands rather than type them. This is not just about aesthetic improvements; it is a step towards creating an app that is more inclusive and accessible, accommodating a variety of abilities and preferences.

Security improvement

Regular pentest/audits: Due to the application having a low impact on the business model of the company, it is not required to provide a lot of security to the application. Regardless, some security measures should be taken, a good approach is a regular penetration test on the application to ensure that no data can be manipulated without proper access rights. These tests could be performed on a yearly basis or even every two years. Another addition that could be made to the security of the application would be security audits, creating a security policy for the application, to remove the potential liability of C-level executives. These could be revised and edited every 6-12 months, considering that after initial creation they would not require a lot of work, it would be a good improvement to the overall structure and also make sure that the application is compliant with necessary legal regulations

Updates to encryption: The application is currently using only Firebase authentication to store passwords, but other user data is not encrypted in any shape or form. Due to this potential data leaks may occur. In order to prevent this, additional encryption methods may be put in place. This would allow the application to be in compliance with all regulatory rules and be more secure overall.

Adding maximum request filter: Though Firebase has one by default, an additional filter against brute forcing a user account may be added. This would stop attackers from cracking passwords just by attempting different wordlists.

Adding a password strength check: The application currently does not contain a password strength check, this allows users to create weak passwords that could be very simple to crack

either by previously mentioned brute forcing attacks or by social engineering attacks revealing relevant information for a password crack.

Adding a SAST scanner to the CI/CD Pipeline: By adding a Static Application Security Scanner (SAST) scanner to the CI/CD pipeline, the application code and dependencies would be cross-checked with a database of known vulnerabilities and the pipeline would fail if something came up as true. This would prevent developers from pushing vulnerable and/or outdated dependencies into the repository and therefore production. Automatically the risk of the application being successfully attacked goes down.

Adding proper security rules to Firebase: The current security rules in the database have been created just as a way to prevent anyone with an API link from accessing the database, but this can and should be further managed to which users should be able to access which collections and documents and with which rights. Proper security Firebase rules would prevent attackers from reading data that should not be read and prevent them from altering it or pushing new data that might be malicious into the database. The current API key for image generation and translation is also stored in cleartext in the dart code. For this, the Firebase secret vault can be used and this would prevent the attacker from sending requests to the given API and potentially causing the company additional costs.

Compliance and Regulatory updates

Finding regulations that apply: The first thing to be done would be to find all the regulations that would apply to the company using the application (based on location of operation and type of application). After this is performed an initial check should be performed to make sure that the current version of the application is compliant with the respective regulations. In case that it is not, it should not be used until the moment that it is and it fulfills all regulatory requirements

Automatic Scanner: Considering that regulatory laws and regulations get updated very often in the EU. There should be put in place an automatic scanner, which will, scan the code for compliance and regulatory issues and provide a full report to the person running the scan.

Scheduled Checks: Assuming that the company running the application does not have a security and legal team looking over the new regulations on the daily to keep everything up to date, there should be scheduled quarterly checks. During these checks, the company should check for potential updates in the regulatory laws and other regulations that apply (i.e.

GDPR). And make sure that the application is fulfilling all respective regulations. In case this isn't done, the application may cause legal issues for the company that is using it.

Triggered Checks: In case the company decides to move/expand to a new location, the local regulations and requirements should be taken into account and it should be made sure that the application is fulfilling these (at least the version running in these places), since the company using the application may get into legal issues in the country of operation if this check is not properly performed (i.e. NIST in the US).

References

- “Build Apps for Any Screen.” Flutter, flutter.dev/. Accessed 4 Oct. 2023.
- CANVA: Visual Suite for Everyone, www.canva.com/. Accessed 30 Oct. 2023.
- “Chatgpt.” ChatGPT, openai.com/chatgpt. Accessed 4 Nov. 2023.
- “The Collaborative Interface Design Tool.” Figma, www.figma.com/. Accessed 4 Oct. 2023.
- “Dall·E 2.” DALL·E 2, openai.com/dall-e-2/. Accessed 15 Oct. 2023.
- Google, Google, firebase.google.com/. Accessed 4 Oct. 2023.
- “Jira.” Agile Board - Jira, [username].atlassian.net/jira/software/projects/SCRUM/boards/1/timeline. Accessed 30 Oct. 2023.
- “Startseite eXXcelent Solutions GmbH.” eXXcellent Solutions, 5 July 2023, www.exxcellent.de/.
- “Where Developers Learn, Share, & Build Careers.” Stack Overflow, stackoverflow.com/. Accessed 2023.

Appendix

I. **Preliminary Design to the customer**

II. **Example of our Sprint Review**

III. **Installation guide**

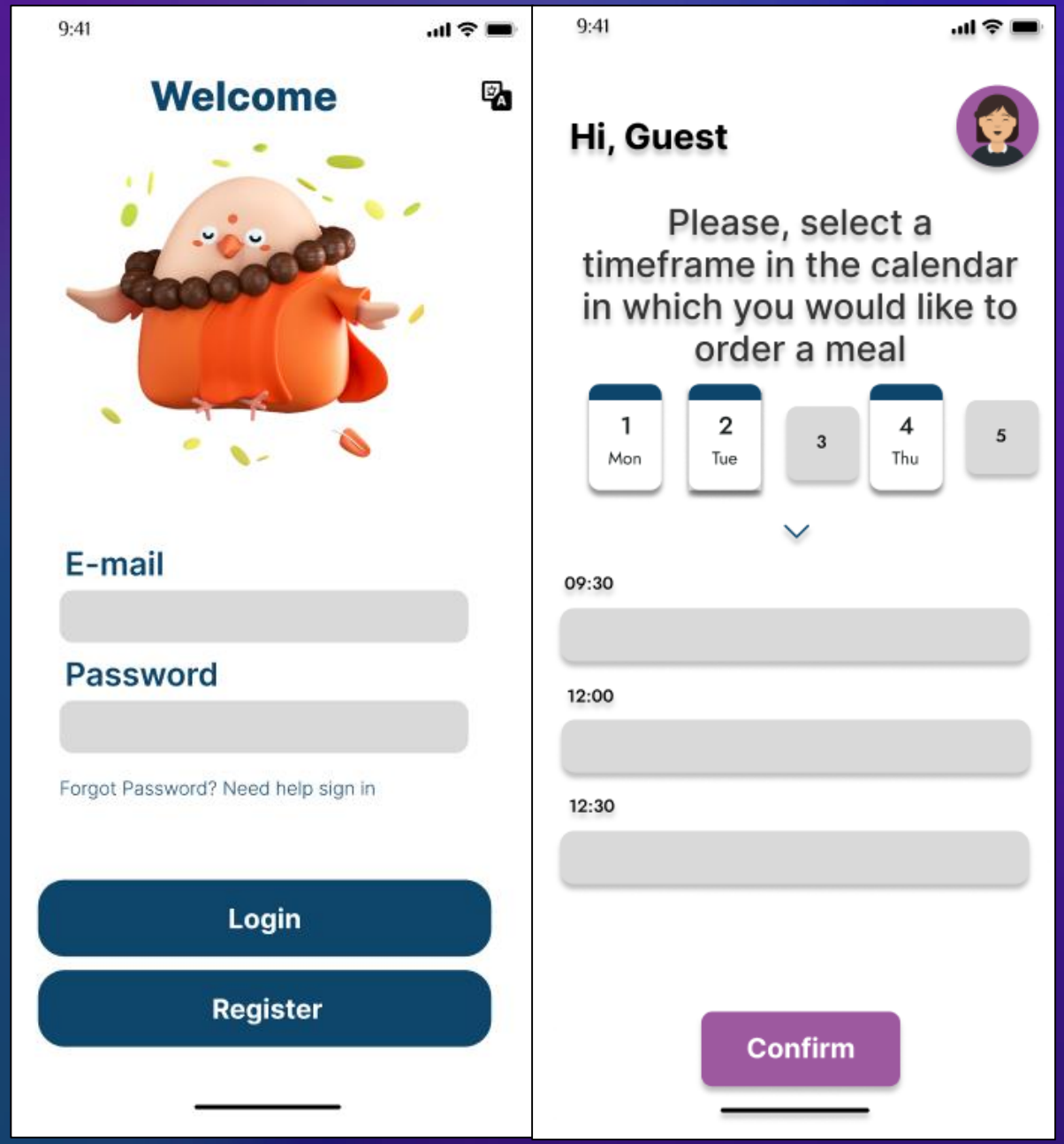
IV. **Poster – Version 1 and 2**

V. **Presentation**

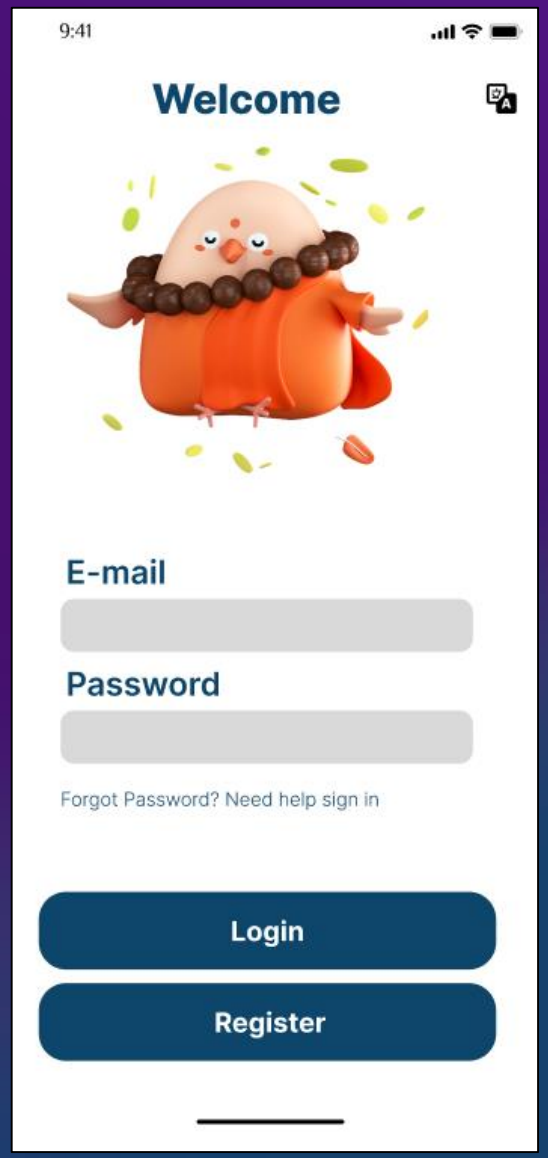
PRODUCT DESIGN

UNVEILING THE DESIGN!

Sprint 1



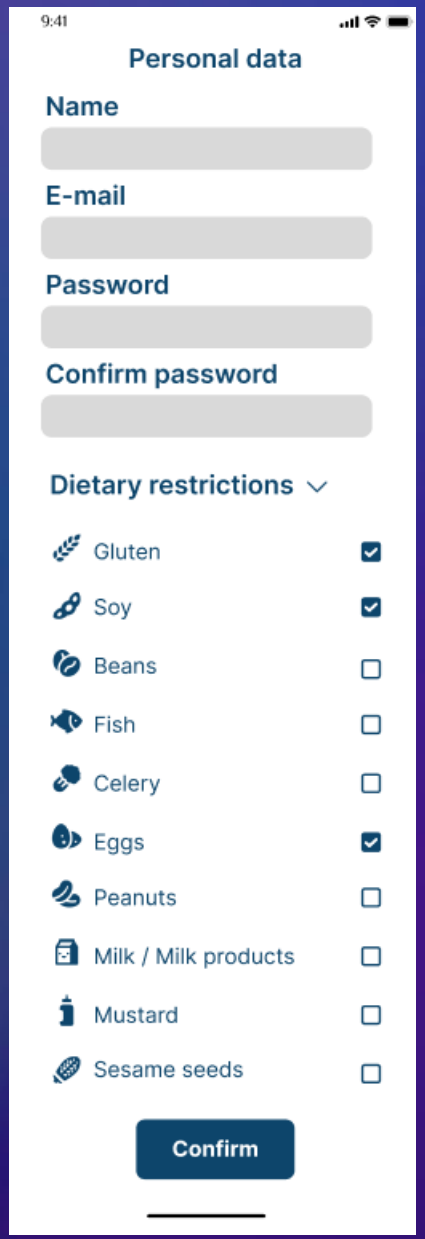
LOGIN/REGISTRATION PAGE



Here the user will be able to login or register themselves in the application.

LOGIN/REGISTRATION PAGE

If the user clicks register, the user will be directed to this screen to enter the preferences. These preferences can be changed later in the 'My Profile' section.



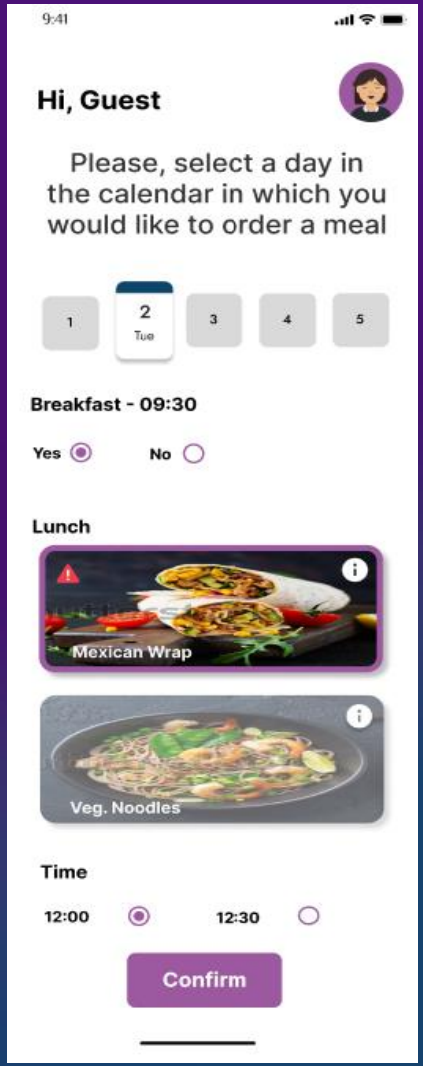
DEFAULT VIEW

1 Selecting the date

2 Filling in the meal choice

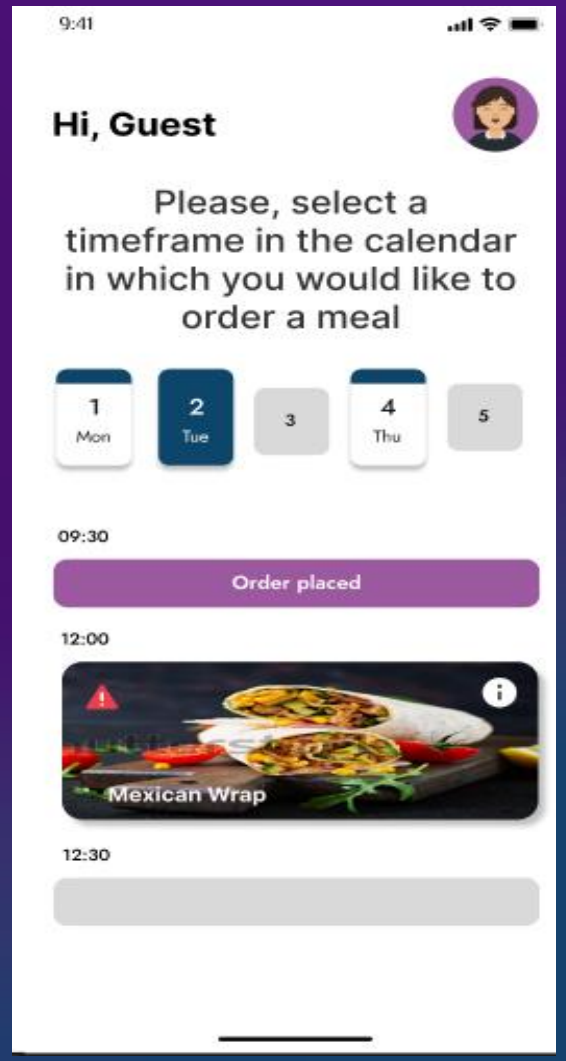
3 Confirming to save your selection

ORDER VIEW

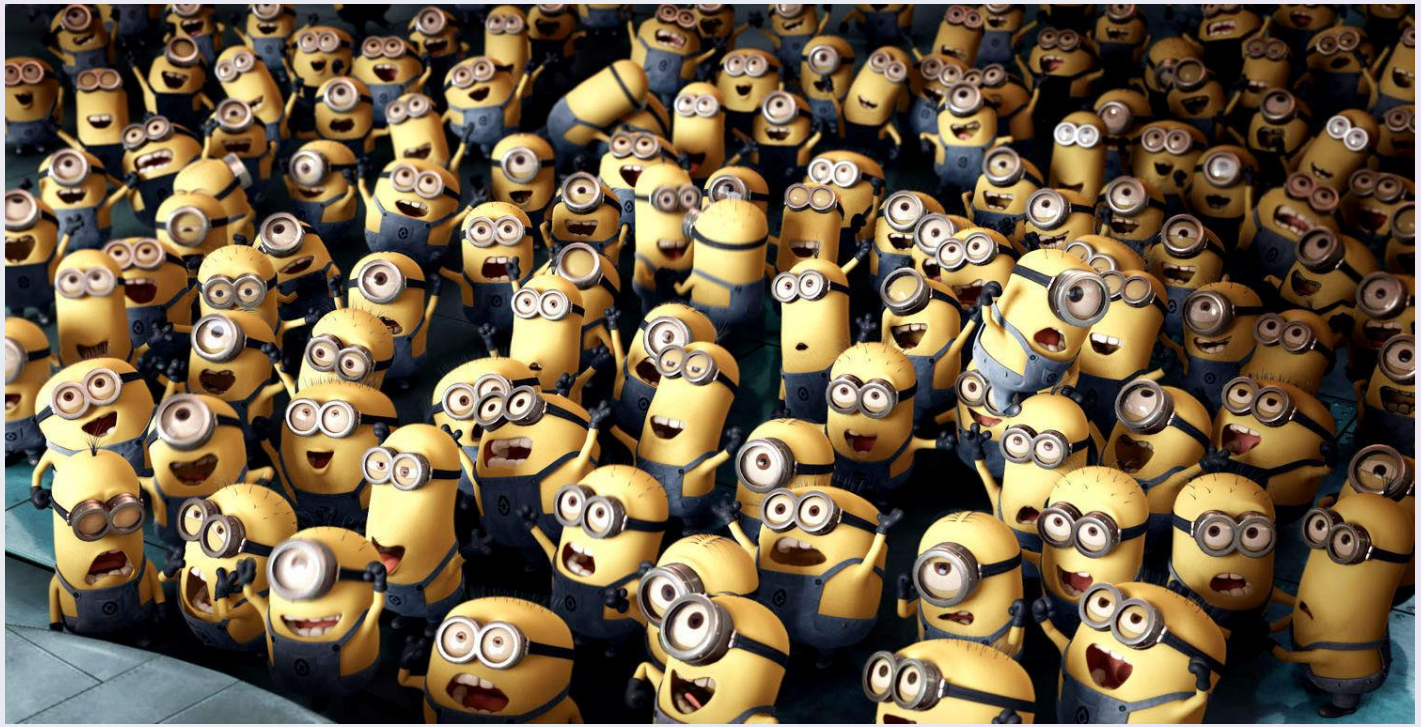


- The user will make the meal selection here

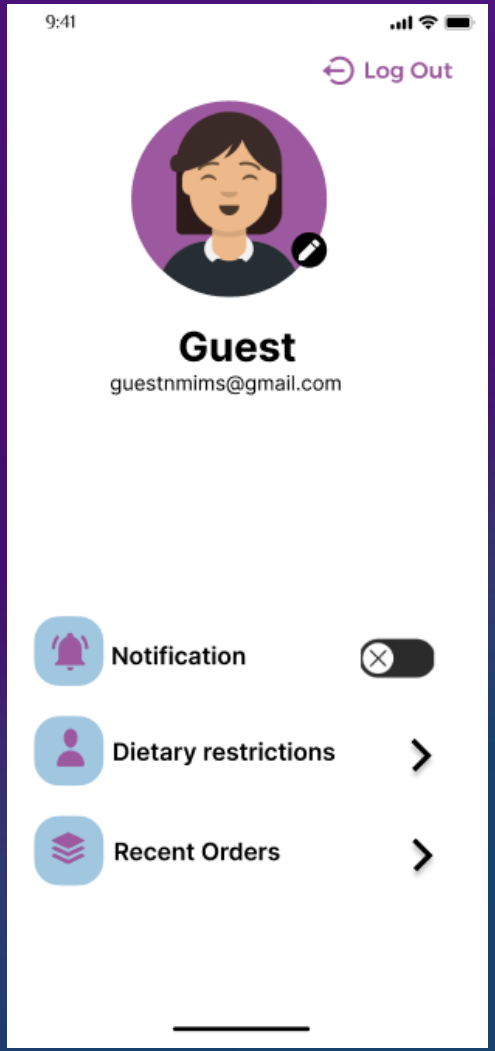
FINAL VIEW AFTER ORDER



- Order Placed!!! Yippeeee!



PROFILE VIEW



- The user will be able to access past orders, edit Dietary restrictions and toggle notifications on or off.

CHEF'S VIEW

CHEF MAIN SCREEN

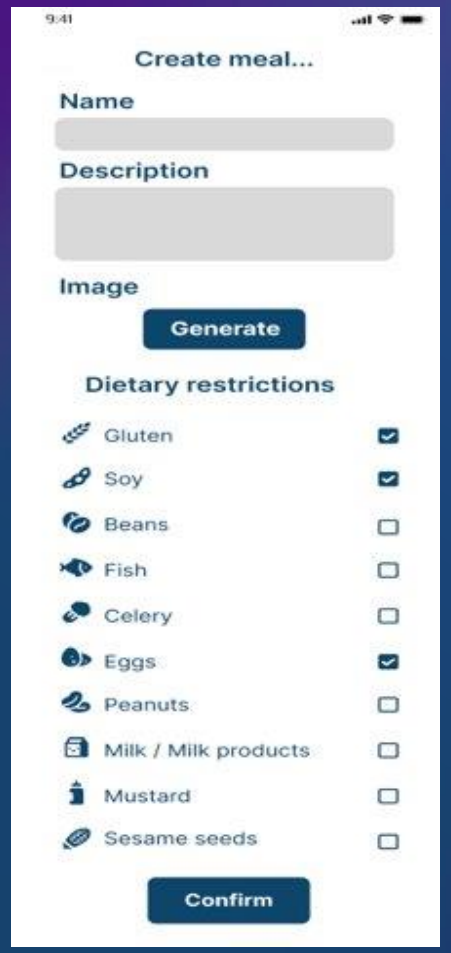
1. Click on the create meal button

2. Write the name and the description of the meal

3. Confirm to save the meal after all the information has been filled

CREATE MEAL

9:41
Create a dish
Create a menu
My Dishes


9:41
Create meal...
Name
Description
Image
Generate
Dietary restrictions
Gluten
Soy
Beans
Fish
Celery
Eggs
Peanuts
Milk / Milk products
Mustard
Sesame seeds
Confirm

9:41
Create meal...
Name
Description
Image
Regenerate
Dietary restrictions
Gluten
Soy
Beans
Fish
Celery
Eggs
Peanuts
Milk / Milk products
Mustard
Sesame seeds
Confirm

Backend:
- Setting up of the database
- Retrieval of the data from firebase
- Authentication , Registration, login, Confirmation email, User verified grant access, filter -> gmail.com

Frontend:
- Design finished for sprint 1
- Registration page
- Menu -> Making the connection with backend

Limitation:
- Only one person

Retrospective:

Marin and Savo:

Notes:
- Mon/Tue too close together in the office.

Issues:
- Spent 3 days helping everyone set up their environment.
- Spent 1.5 days fixing firebase.
- Working on old code.
- No necessary equipment at the company premise.
- Too many middle men (Chain of command) - Ask Prof. Graf if anything can be done
- random meetings

Went well:
- Authentication
- Data retrieval from firestore
- Meetings in Person (Customer and internal)
- Amount of work done alongside Savo
-

Improvements:
- Make the authentication better
- Get snacks during work meetings
- Do a better job in databases diagrams
- We need more monitors - Talk to Mrs. Bosch

Sophia:

Notes:
-

Issues:
- Problems with Flutter

- Programming problems with Flutter(Half Day)
- Problems with the emulator
- Figuring out how to work with Figma (Beginning)

Went well:
- Programming of the registration page
- Design

Improvements:
- Next week: Program the pages faster
- Fix the design

Vasiliy:

Notes:
-

Issues:
- Communication - random Meetings - Receive the invitation in advance.
- thinks that he will struggle with IOS

Went well:
- Design

Improvements:
- Snacks
- Monitors

Yatharth:

Notes:
- Have dailys to snyc

Issues:
- Spending 3 days to set up the environment
- CPU Speed
- Meetings popped up randomly (My bad)
- Group meeting timings
- Time slots dont line
- Communication (First half week)

Went well:
- Good team spirit
- Helpfulness of the team members
- Creation of the documentation, product design presentation
- Communications (Second half week)

Improvements:

- Communication - Use one shared communication method - Google calendar
- Snacks
- Breaks, Team Building excercises

Retrospective feedback:

- Doing it together
- No separate naming
- Single improvement sections

Review Feedback:

- Look at the tasks top down
- Scrum board up to date ? - User story wise
-

How to install an APK File to you Android Device!

Group Members:

- Savo Simeunovic
- Marin Susic
- Sophia Soares
- Vasiliy Klyosov
- Yatharth Upmanue

What to Know!

- Allow Chrome to install unknown apps by going to **Settings** > **Apps** > **Menu** > **Special access** > **Install unknown apps**.
- Install a file manager (such as Cx File Explorer or File Manager) so that you can find the APK file after you download it to your phone.
- Download an APK file and open it to install it. Alternatively, transfer the APK Installer from your computer using USB.
- If you want to use an app from outside of the Google Play Store, you can install the app's APK file. This article explains how to install APKs on Android 7 or later.

Allow Unknown Apps on Android

Before you can download APK files using Chrome or any other browser, you must first allow unknown apps:

- Go to your device settings and tap **Apps & Notifications** (or **Apps** in older versions of Android).
- Tap the **three dots** in the upper-right corner.
- Tap **Special access**.

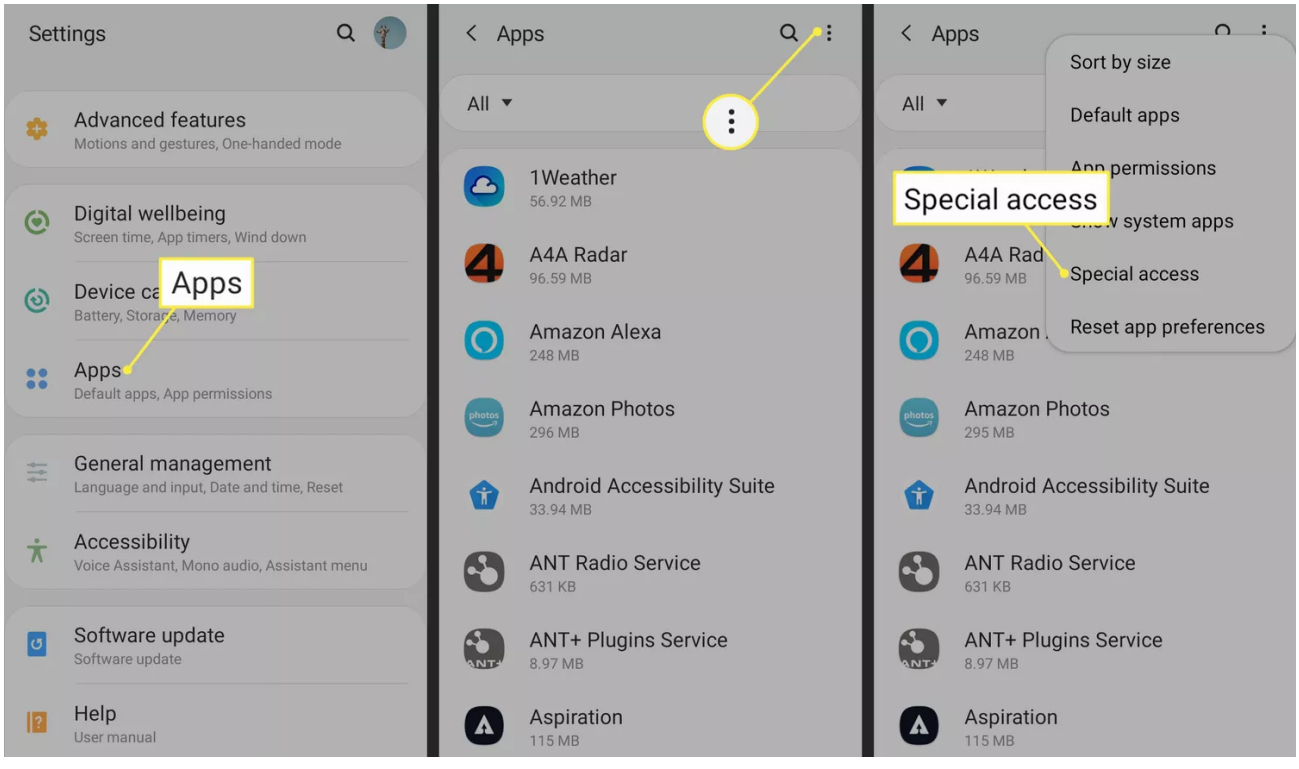


Image Source: lifewire

- Tap **Install unknown apps**.
- Tap **Chrome** (or whichever web browser you use)
- Move **Allow from this source** to the **On** position.

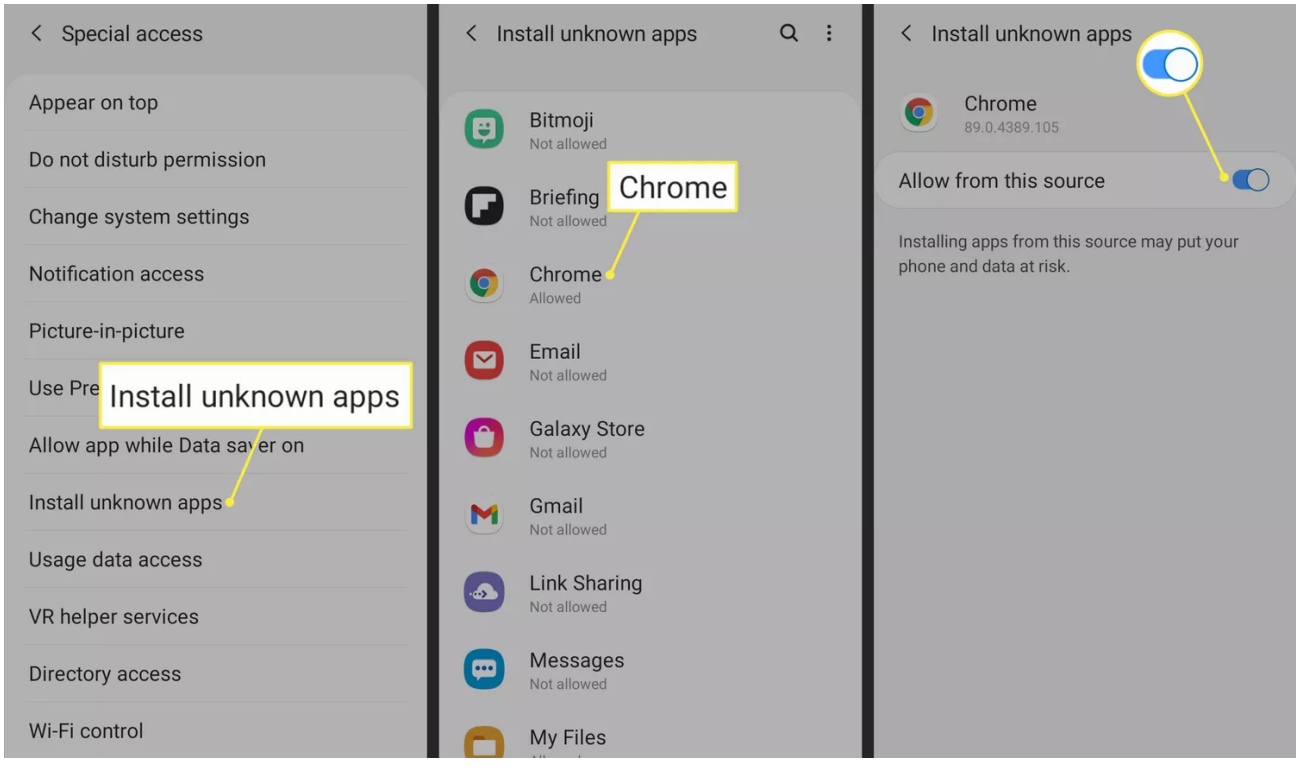


Image Source: lifewire

Install an Android File Manager

Now that you've configured your phone to let you install unknown apps, you'll need a way to find the application file (APK file) on your phone so that you can run it.

Android phones typically come with a file manager app you can use, but if you don't have one, search for one on Google Play.

Download the APK Installer From Your Android

The easiest way to install an APK file on your Android is to download the file using the default browser, Chrome.

1. Tap the link to download the APK file. Accept any pop-ups, including “This type of file can harm your device.”
2. If your phone's web browser doesn't give you the option to open the file after downloading, open your file explorer app, go to the **Downloads** folder on your device, then tap the **APK file**.
3. Allow the app any required permissions it asks for. Then, at the bottom of the installer window, tap **Install**.
4. Now you'll see the app available in your list of installed apps.

Transfer the APK Installer via USB

If you don't have internet access on your phone, or for any other reason you can't use a browser to download the file, you can still install apps from your computer. Using your computer web browser, download the APK file just like described above. Once it's downloaded, connect to your Android phone and transfer the file.

If you've never connected your Android to your computer, turn on USB debugging mode on Android. You can then plug your phone into your computer with a USB cable, and it'll mount the phone like it would a memory stick.

Once your phone is connected to your computer, it shows up as another drive in your computer's File Explorer. Move the APK file you downloaded from the app website to the **/sdcard/download** folder on your phone.

Once the file is transferred, use the file explorer app on your phone as described in the previous section to tap the APK file and install the app.

If you don't have a USB cable, another solution is to install WiFi FTP Server from Google Play. Then, use a free FTP client software program on your computer, to transfer the APK file from your computer to the **/sdcard/download** folder on your phone. However, this is an advanced option and requires an understanding of how to use FTP files.

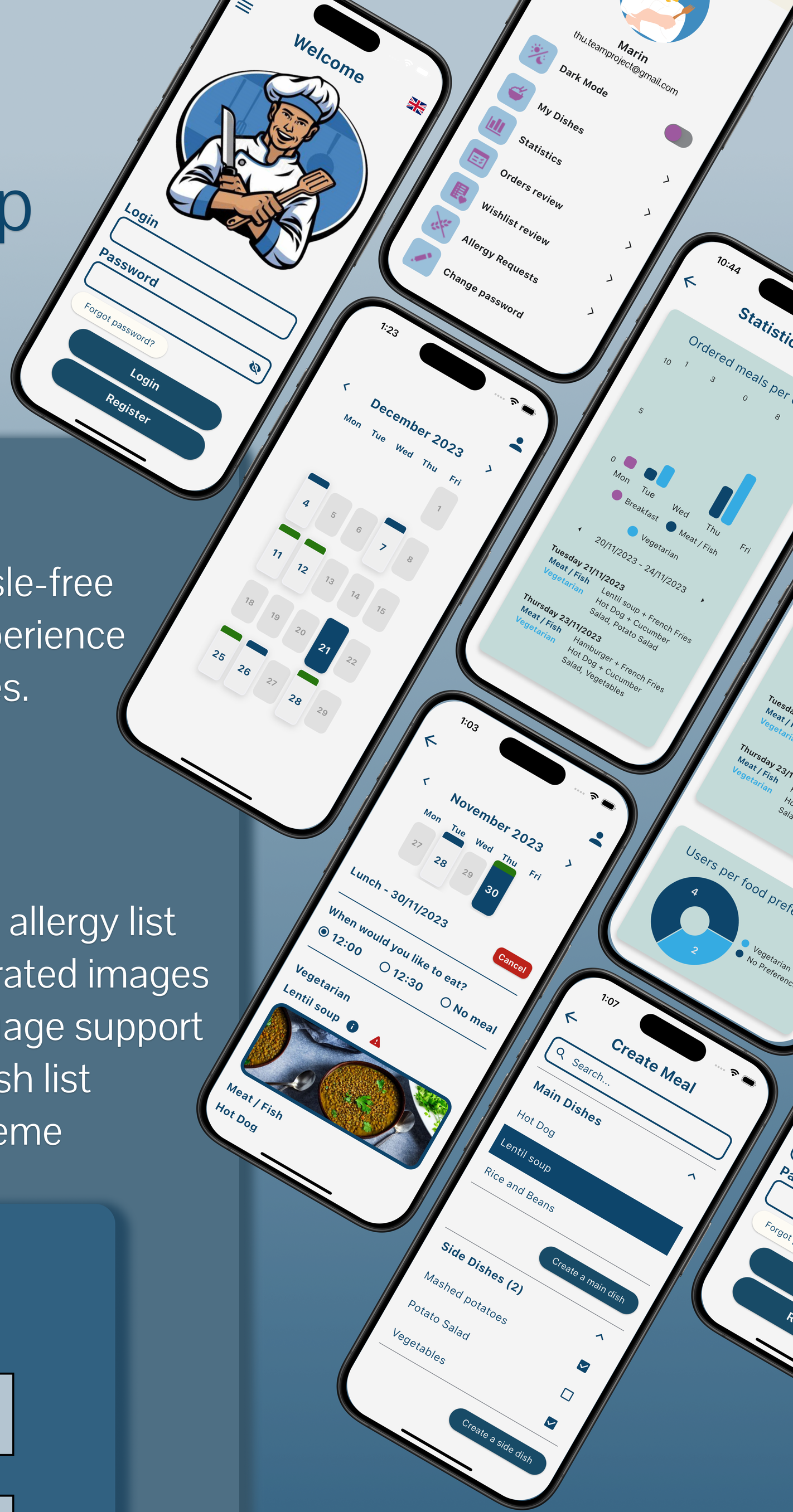
Source: How to install APK on Android Minutes.com

THU Technische Hochschule Ulm University of Applied Sciences

ex|Xcellent solutions

Menu Ordering App

Team-oriented project
Winter Semester 2023/24



Project Goal

To provide our users with a hassle-free and enjoyable food ordering experience using enhanced Kitchen features.

Features

- Meal ordering
- Meal rating
- Notifications
- Statistics
- Dish management
- Custom allergy list
- AI generated images
- 15 language support
- Meal wish list
- Dark theme

Architecture

view (screens, widgets, providers)
viewmodel
service
rest: translating API, image generating API, / database (Firebase Firestore)

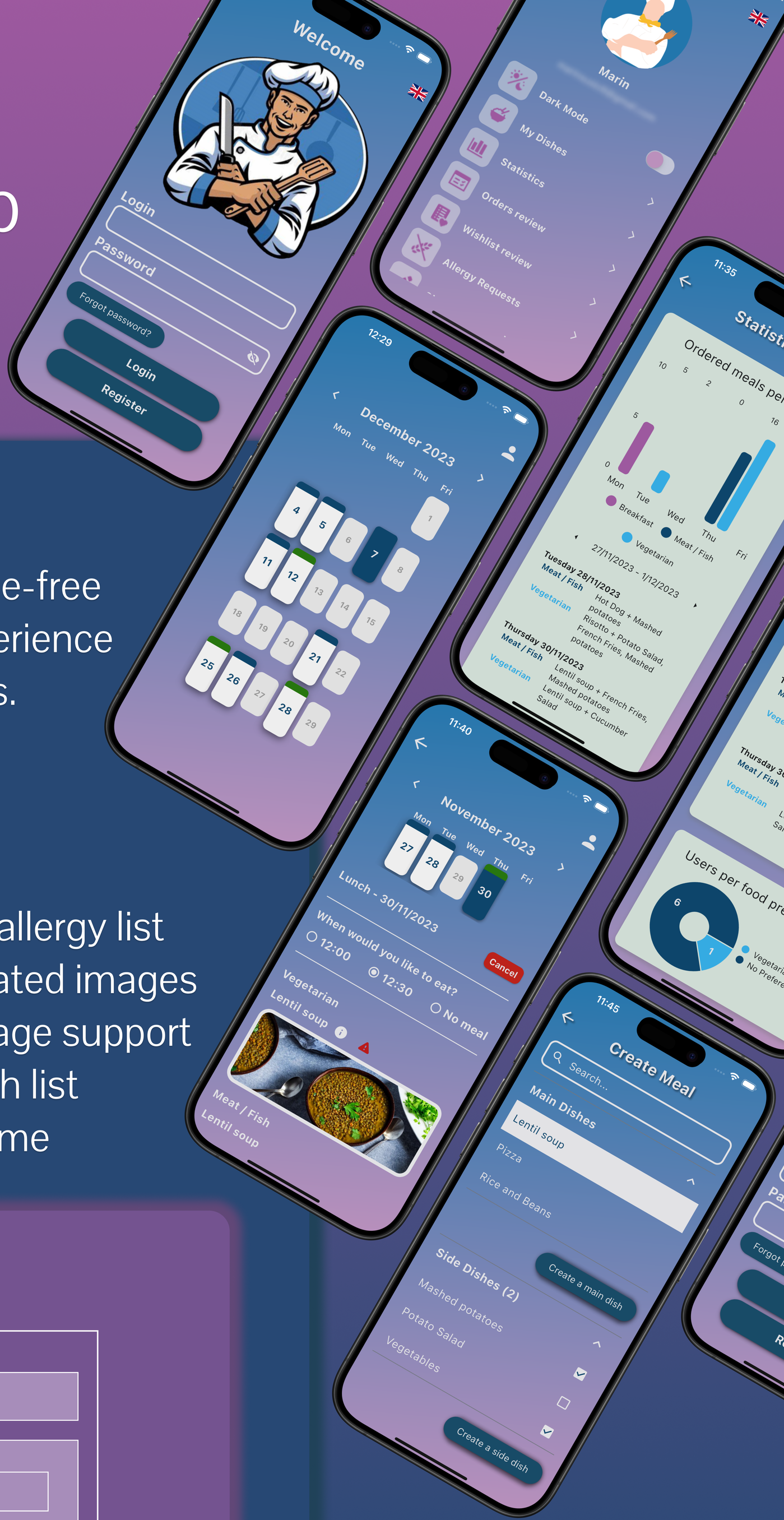
Marin Sušić
Savo Simeunović
Sophia Soares
Vasiliy Klyosov
Yatharth Upmanue

THU Technische Hochschule Ulm University of Applied Sciences

exXcellent solutions

Menu Ordering App

Team-oriented project
Winter Semester 2023/24



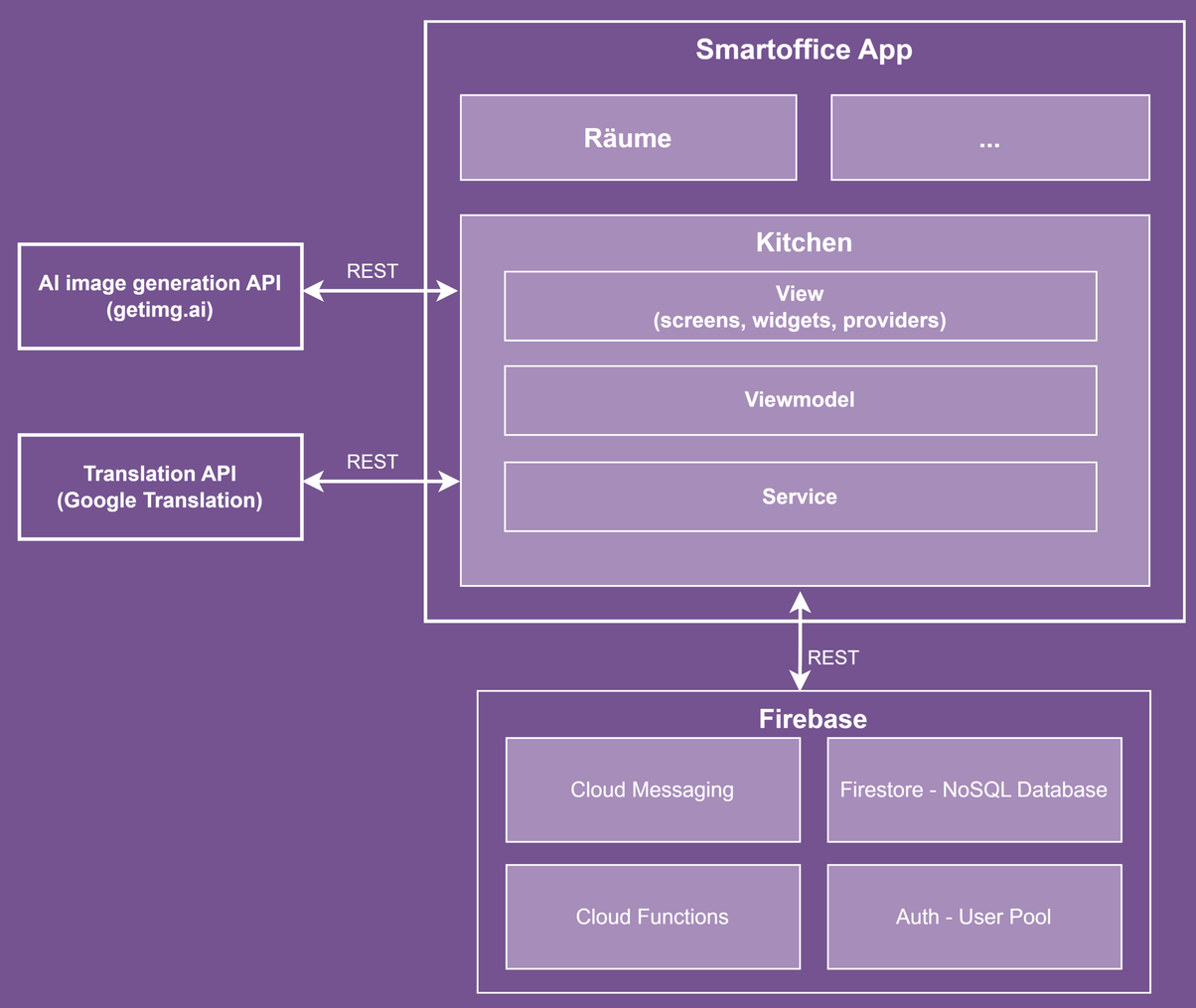
Project Goal

To provide our users with a hassle-free and enjoyable food ordering experience using enhanced Kitchen features.

Features

- Meal ordering
- Meal rating
- Notifications
- Statistics
- Dish management
- Custom allergy list
- AI generated images
- 15 language support
- Meal wish list
- Dark theme

Architecture



Marin Sušić
Savo Simeunović
Sophia Soares
Vasiliy Klyosov
Yatharth Upmanue

ex|Xcellent solutions

THU Technische Hochschule Ulm University of Applied Sciences

Team 6

Smart Office App - Kitchen Module

Our Team

Sophia Soares

Developer

Marin Sušić

Developer

Savo Simeunović

Developer

Vasiliy Klyosov

Developer

Yatharth Upmanue

Project Manager / Developer

Introduction

The Smartoffice app is intended to provide employees of eXXcellent solutions GmbH with:

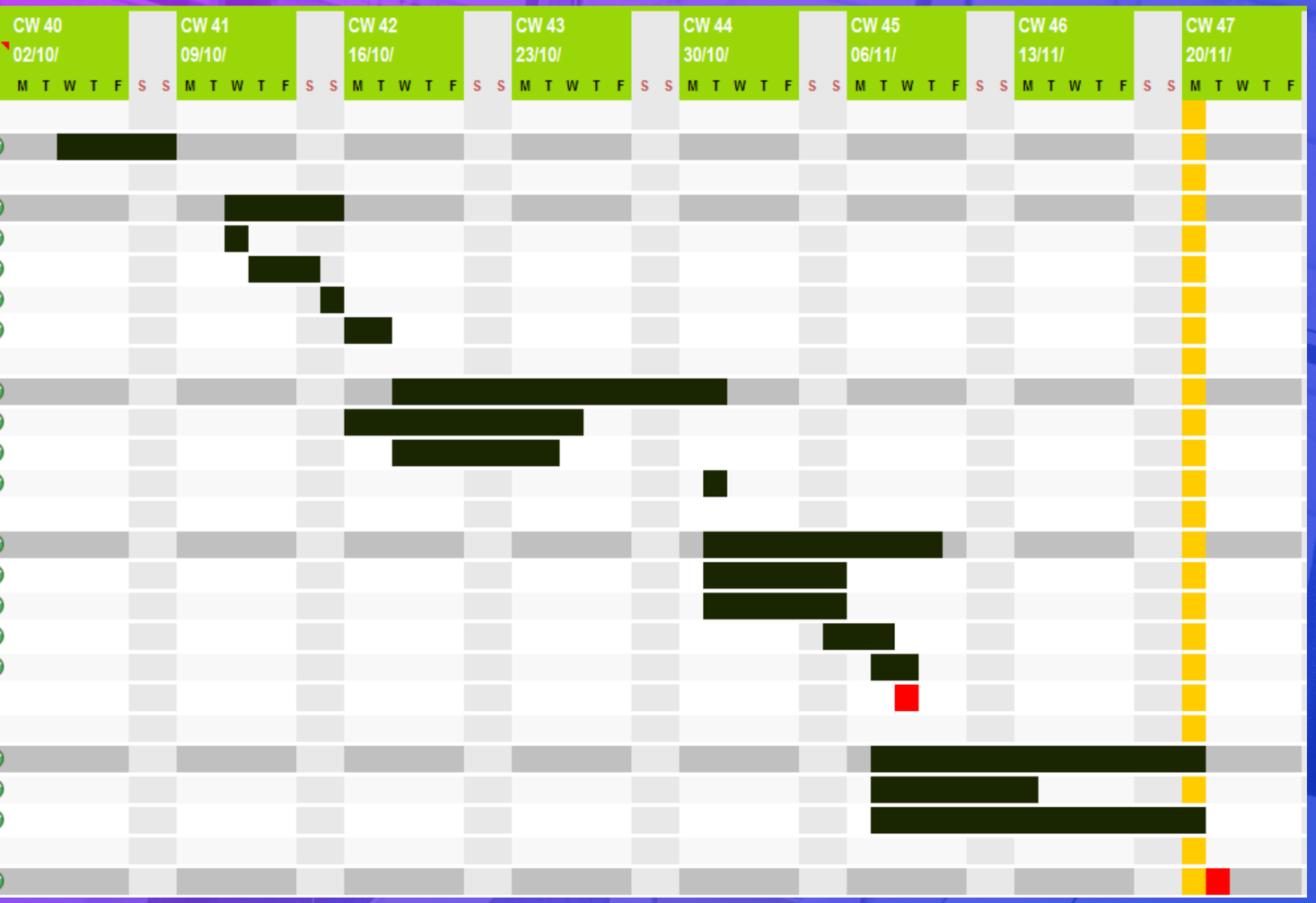
- Convenient access to functionalities:
  - Room Heating
  - Parking
  - Assistant
  - **Kitchen**

Kitchen Ordering Module

- Focus on food ordering functionality
- Simplifying and streamlining the meal preparation process
- Efficient Breakfast and Lunch Ordering
- Enhanced Kitchen Features
- Hassle-free
- Empower the employees

Project Plan

Tasks	Who	Start	Duration	End	Status
Kick-Off		04/10/23	5 D	08/10/23	100%
Concept		11/10/23	5 D	15/10/23	100%
Customer Requirements Finalization		11/10/23	1 D	11/10/23	100%
Preliminary Design		12/10/23	3 D	14/10/23	100%
Backend Structure		15/10/23	1 D	15/10/23	100%
Database Structure		16/10/23	2 D	17/10/23	100%
Implementation Phase 1		18/10/23	14 D	31/10/23	100%
Creating Frontend Programming structure		16/10/23	10 D	25/10/23	100%
Creating Backend basic structure		18/10/23	7 D	24/10/23	100%
Merging Backend and Frontend		31/10/23	1 D	31/10/23	100%
Implementation Phase 2		31/10/23	10 D	09/11/23	100%
Enhancing Backend structure for optional Features		31/10/23	6 D	05/11/23	100%
Enhancing Frontend structure for optional Features		31/10/23	6 D	05/11/23	100%
Working on Customer feedback		05/11/23	3 D	07/11/23	100%
Final Internal Testing and Bugs removal		07/11/23	2 D	08/11/23	100%
Feature Freeze		08/11/23	M		
Control & Monitoring		07/11/23	14 D	20/11/23	100%
Testing		07/11/23	7 D	13/11/23	100%
Review, Modification and final bug fixes		07/11/23	14 D	20/11/23	100%
App-GoLive - Final Presentation in the company		21/11/23	M		



Required Features

Chef

Menu management

- Create, update, and delete: breakfast and lunch

Meal Rating

- See rating
- See feedback

Statistics

- No. of people
- No. of veg. meals
- No. of non-veg. meals

Notification

- Food is ready
- Changes in menu

Employee

Meal Ordering

- Order breakfast
- Order lunch
- Invite guests

Meal Rating

- Star rating
- Feedback for Chef
- See past meals

Notification

- Food is ready
- Changes in menu
- Order expiration

Meal Cancellation

- Meal can be cancelled

Optional Features

1

Statistics:

- Employees who are veg
- Employees who are non-veg

2

Notifications:

- Food Buzzer: food is still left!

3

AI Generated Images:

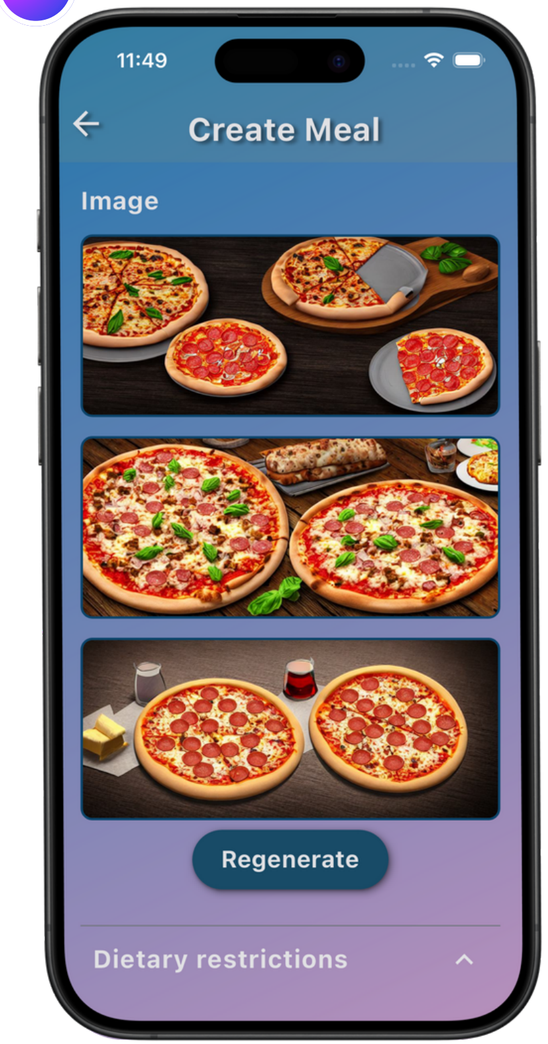
- Customized images for each meal

4

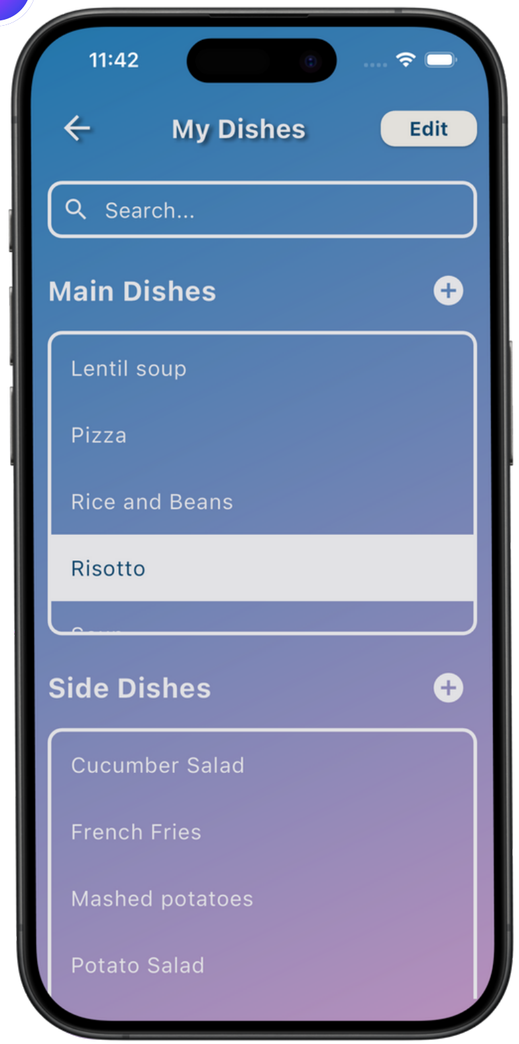
Management of Dishes:

- Add, delete and update main and side dishes

3



4



Optional Features

5

Allergy List:

- Default allergies provided
- Allergy warning

6

Custom Allergy List:

- Employees can suggest custom allergies

7

Multiple Language and Dark Theme Support

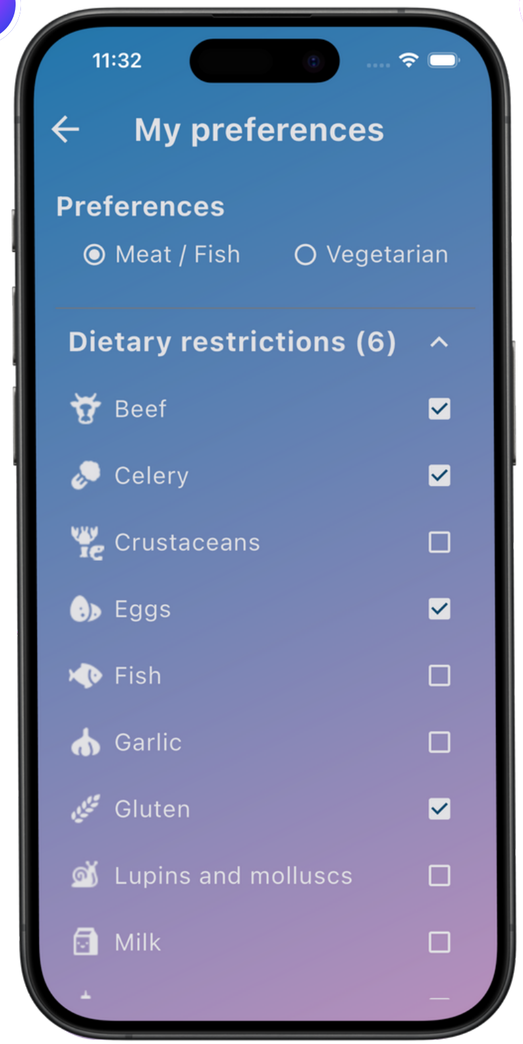
- 15 Languages
- Users can use the app in dark mode

8

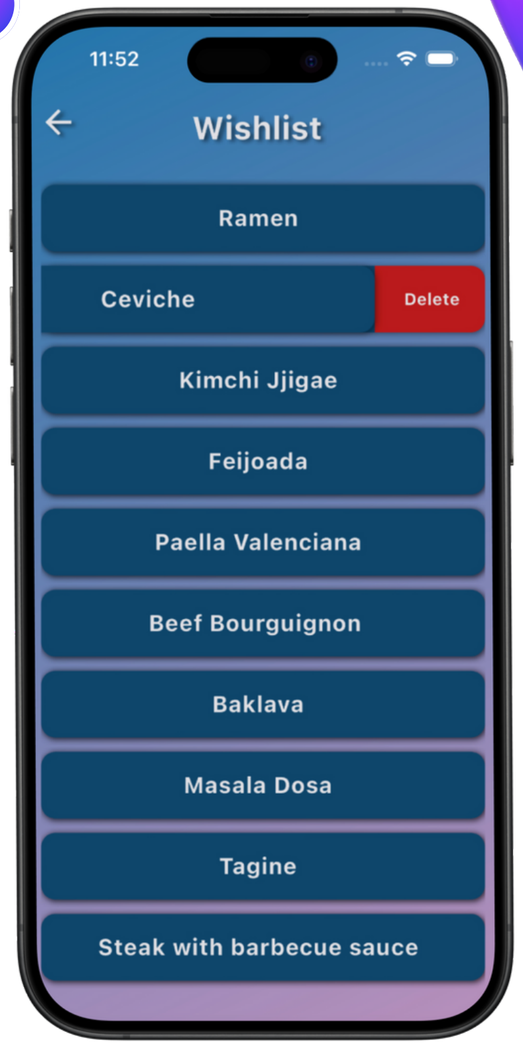
Food Wish List:

- Employees can suggest dishes
- Wishlist view

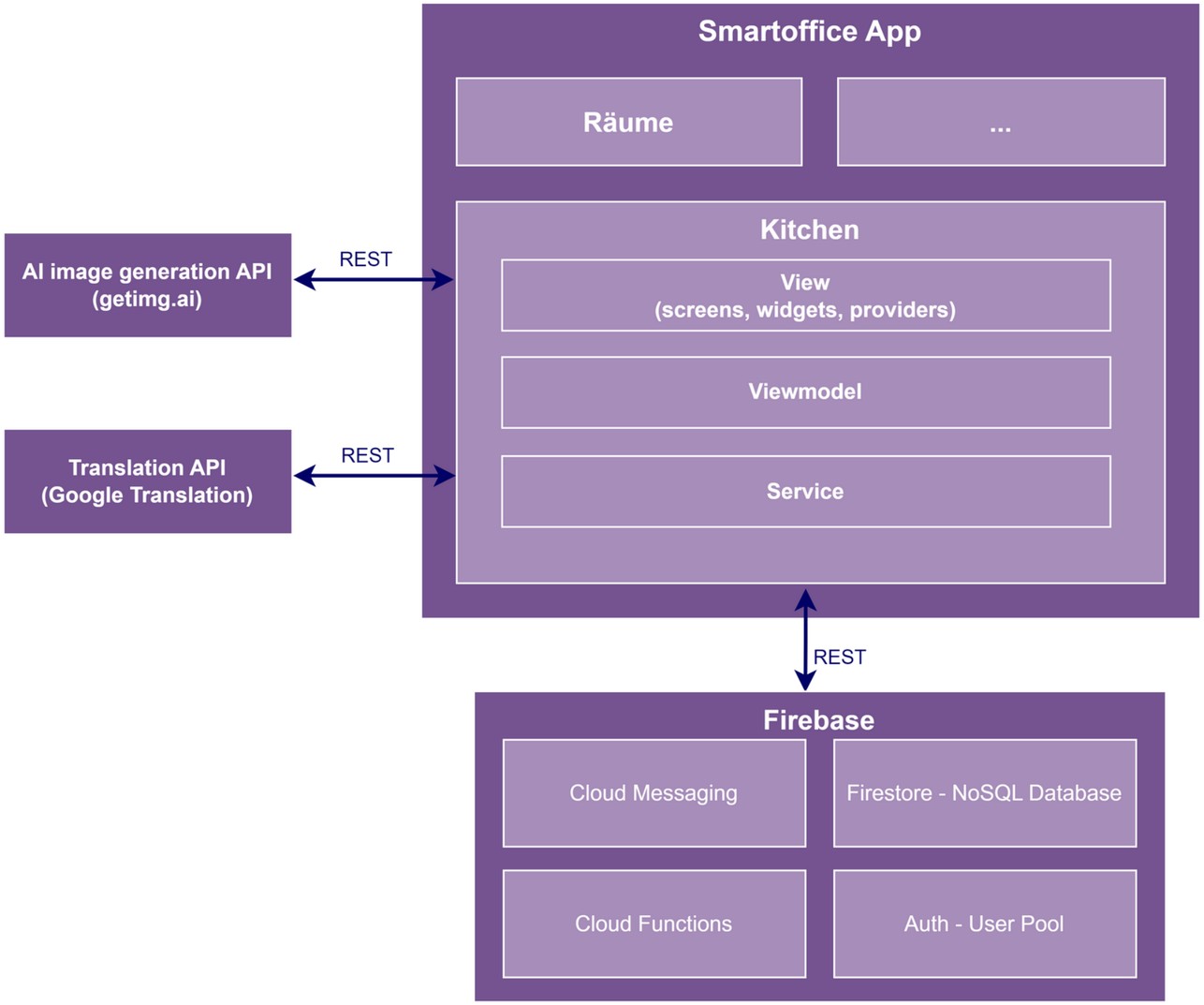
5



8

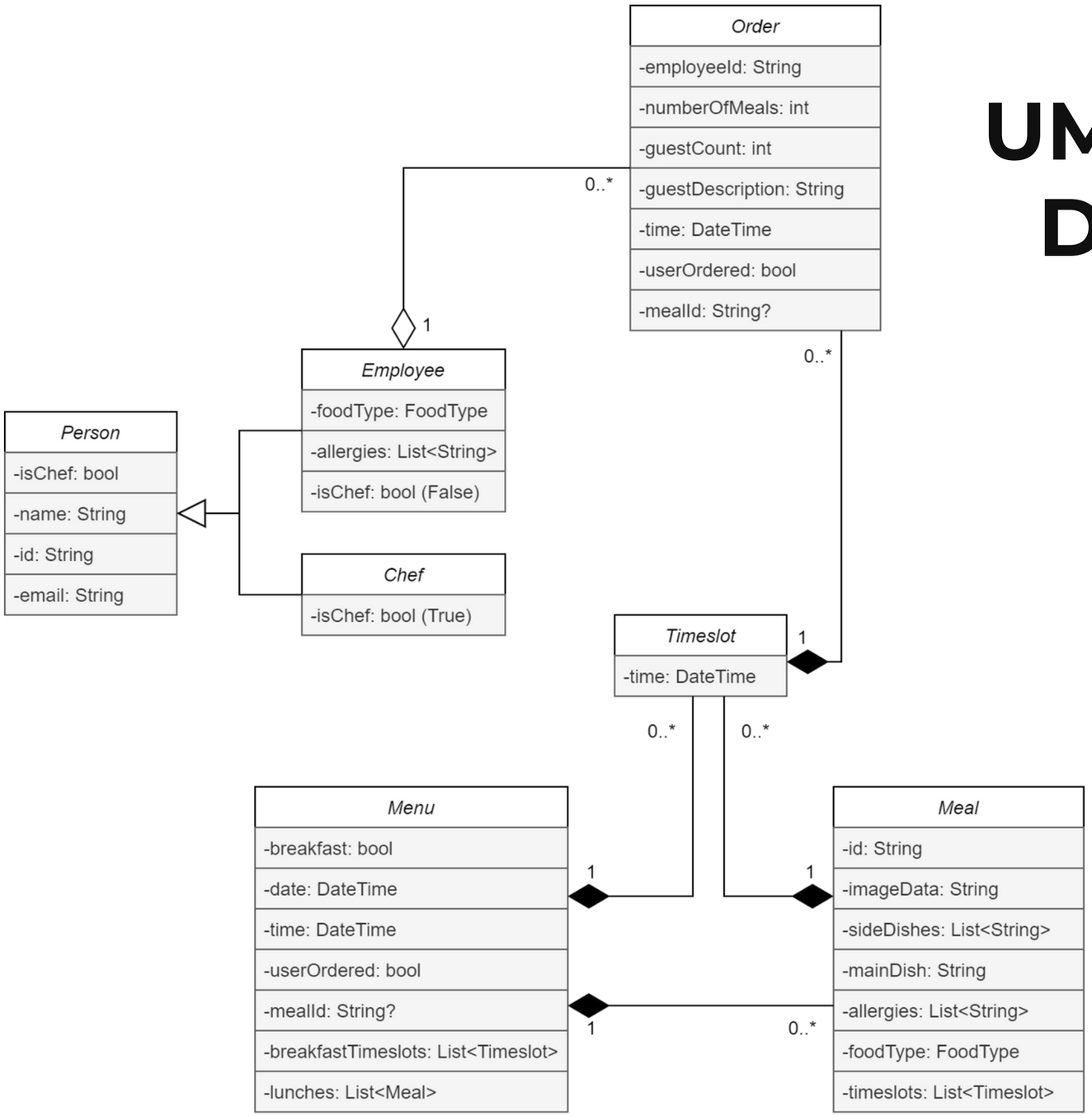


Technical Details



Application structure

UML Class Diagram



Challenges

Challenge 1

Working with previously created code

Challenge 2

Working with deprecated software

Challenge 3

Connecting issues between the previous and new code

Challenge 4

Establishing and connecting firebase with the previous code

Challenge 5

Getting IOS Application to run after small changes in code

DEMO

Lessons Learned

- Unexpected issues can pop up at any time
- Team work requires patience
- Importance of regular testing
- Effective communication is key

Outlook

- Redirecting the chef from the statistics to the order review
- Redirecting the chef from the wish list to the meal creation
- Chef can add surprise dishes
- Using AI to generate recipes

Customer Feedback

- “The students **started the project professionally** and with commitment from the very first minute.”

- “I was particularly **impressed** by the **students' independent way of working**. We only had to act as Customers”

- “Their **project plan was well thought out and tailored to the customer's satisfaction**
  - a functional prototype after 4 weeks for testing,
  - a professional presentation of the prototype to the customer including known features
  - **actively obtaining feedback** and enough time for bug-fixing
  - **implementing customer's feedback**).”

- “The students **implemented far more functionalities** than expected, delivered **good code quality**, thought about the **appropriate architecture** and wrote **quality-assuring tests**.”

Team 6

Thank you

Questions?