****

**Software Project Management Plan**

**foodo**

**Team 11**

**Alper Çelik, 21903022**

**Batuhan Duras, 22002854**

**Baturalp Sönmez, 22103933**

**Cemal Fırat Dağ, 22003819**

**Davut Durmaz, 21903291**

Bilkent University

Department of Information Systems and Technologies

08.12.2024

Change History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File Name** | **Document Type** | **Deliverable** | **Version** | **Submission Date** |
| Deliverable\_3.docx | Microsoft 365 | 3 | 0.1 | - |
| Deliverable\_3\_1.docx | Microsoft 365 | 3 | 0.2 | - |
| Deliverable\_3\_2.docx | Microsoft 365 | 3 | 0.3 | - |
| Deliverable\_3\_3.docx | Microsoft 365 | 3 | 0.4 | - |
| Deliverable\_3\_4.docx | Microsoft 365 | 3 | 0.5 | - |
| Team11\_SPMP\_FR1.docx | Microsoft 365 | 3 | 1 | 08.12.2024 |

Project Team

|  |  |  |
| --- | --- | --- |
| **Name, Surname** | **Student Id Number** | **e-mail** |
| Alper Çelik | 21903022 | alper.celik@ug.bilkent.edu.tr |
| Batuhan Duras | 22002854 | batuhan.duras@ug.bilkent.edu.tr |
| Baturalp Sönmez | 22103933 | baturalp.sonmez@ug.bilkent.edu.tr |
| Cemal Fırat Dağ | 22003819 | firat.dag@ug.bilkent.edu.tr |
| Davut Durmaz | 21903291 | davut.durmaz@ug.bilkent.edu.tr |

Project Details

|  |  |
| --- | --- |
| **Project Name** | **foodo** |
| **Software Name** | **foodo** |
| **Company Name** | **No sponsor company** |
| **Academic Advisor** | **Dr. Serkan Genç** |
| **GitHub URL** | **https://github.com/foodoHub** |
| **WEB page** | **https://foodo.cfd** |

Individual Contributions Overview

|  |  |
| --- | --- |
| **Name, Surname** | **Summary of Contributions to the Initial Plan Document** |
| Alper Çelik | Contributed to the whole document on equal terms as his teammates. |
| Batuhan Duras | Contributed to the whole document on equal terms as his teammates. |
| Baturalp Sönmez | Contributed to the whole document on equal terms as his teammates. |
| Cemal Fırat Dağ | Contributed to the whole document on equal terms as his teammates. |

Executive Summary

The Software Project Management Plan (SPMP) provides a comprehensive framework for project management activities, including planning, execution, monitoring, and verification, ensuring the delivery of a high-quality and user-centric application. The foodo project has two primary objectives: to create a personalized cooking assistant that provides tailored meal suggestions and step-by-step cooking guidance, and to establish a vibrant social platform where users can share meal photos, gain achievements, and interact with like-minded food lovers. These features aim to make cooking accessible and enjoyable for users of all skill levels. The SPMP emphasizes detailed planning and effort estimation. Effort was calculated using Work Breakdown Structure (WBS), Use Case-Based Estimation, and Agile Estimation. The most reliable result, derived from Use Case-Based Estimation, projects 2347.02 person-hours for the project. Resource requirements include personal hardware and software tools, cloud-based services like AWS and Google Vertex AI, and collaborative platforms such as Slack and GitHub. A cost analysis and Net Present Value calculation provide financial insights, ensuring efficient budget allocation. The document specifies key metrics and tools for project monitoring, such as task completion rates, feature coverage metrics, and defect resolution times, ensuring alignment with objectives. Verification and validation (V&V) strategies include code reviews, unit testing, API testing, and user acceptance testing. The project leverages a modern microservices architecture to ensure scalability and maintainability. The SPMP also identifies challenges, including time constraints, team coordination, and health impacts, and discusses adaptive strategies such as workload redistribution and switching communication platforms from Discord to Slack. Sustainability and ethical considerations were addressed by minimizing environmental impact through digital tools and ensuring transparency in resource usage. In conclusion, this SPMP reflects Team 11's dedication to creating a platform that makes cooking enjoyable and social. By combining careful planning and adaptability, we are confident in delivering a user-friendly, engaging application that enhances how people cook and connect with others.[[1]](#footnote-1)

Table of Contents

**Page Number**

[Executive Summary i](#_Toc184538200)

[Table of Contents ii](#_Toc184538201)

[1 Scope 1](#_Toc184538202)

[2 Project Schedule 1](#_Toc184538203)

[2.1 Work Packages & Dependencies 1](#_Toc184538204)

[2.2 Resource Requirements 3](#_Toc184538205)

[2.3 Cost Estimation, Net Present Value, Budget and Resource Allocation 7](#_Toc184538206)

[3 Project Effort Estimation 11](#_Toc184538207)

[3.1 Work Breakdown Structure (WBS) - Decomposition-Based Estimation 11](#_Toc184538208)

[3.2 Use Case Based Estimation 13](#_Toc184538209)

[3.3 Agile Estimation 16](#_Toc184538210)

[3.4 Discussion 17](#_Toc184538211)

[4 Project Monitoring and Measuring 19](#_Toc184538212)

[5 Product Verification and Validation 23](#_Toc184538213)

[6 Software Development Environment 35](#_Toc184538214)

[7 Discussions 36](#_Toc184538215)

[7.1 Limitations and Constraints 36](#_Toc184538216)

[7.2 Health and Safety Issues 37](#_Toc184538217)

[7.3 Legal Issues 38](#_Toc184538218)

[7.4 Economic Issues and Constraints 38](#_Toc184538219)

[7.5 Sustainability 38](#_Toc184538220)

[7.6 Ethical Issues 38](#_Toc184538221)

[7.7 Multidisciplinary Collaboration 38](#_Toc184538222)

[8 References 39](#_Toc184538223)

List of Tables

**Page Number**

[Table 1. Employee Cost Estimation 8](#_Toc184538180)

[Table 2. Development Expenses 8](#_Toc184538181)

[Table 3. Cloud and API Costs 9](#_Toc184538182)

[Table 4. NPV Calculation Table 10](#_Toc184538183)

[Table 5. Unadjusted Actor Weights 14](#_Toc184538184)

[Table 6. Unadjusted Use Case Weights 15](#_Toc184538185)

[Table 7. Technical Complexity Factors 15](#_Toc184538186)

[Table 8. Environmental Factors 15](#_Toc184538187)

[Table 9. Story Point Estimation Matrix 16](#_Toc184538188)

[Table 10. User Story Points 17](#_Toc184538189)

[Table 11. Verification and Validation Schedule 26](#_Toc184538190)

[Table 12. Code Review Test Case 27](#_Toc184538191)

[Table 13. Static Code Analysis Test Case 28](#_Toc184538192)

[Table 14. Unit Testing Test Case 29](#_Toc184538193)

[Table 15. API Testing Test Case 30](#_Toc184538194)

[Table 16. UAT Test Case 31](#_Toc184538195)

[Table 17. Compatibility Testing Test Case 32](#_Toc184538196)

[Table 18. Fault Injection Test Case 33](#_Toc184538197)

[Table 19. Usability Testing Test Case 34](#_Toc184538198)

[Table 20. Details of Software Development Environment 35](#_Toc184538199)

List of Figures

[Figure 1. foodo.cfd Order Summary 7](#_Toc184538174)

[Figure 2. Cash Flow Diagram 9](#_Toc184538175)

[Figure 3. UML Use Case Diagram [1] 13](#_Toc184538176)

[Figure 4. GitHub Prioritized Backlog Page of foodo 21](#_Toc184538177)

[Figure 5. GitHub Status Board of foodo 22](#_Toc184538178)

[Figure 6. Receipt for Energy Drinks 37](#_Toc184538179)

Abbreviations

|  |  |
| --- | --- |
| CRUD | Create, Read, Update & Delete |
| EFactor | Environmental Factor Value |
| EF | Environmental Factor |
| NPV | Net Present Value |
| PHM | Person Hour Multiplier |
| SDD | Software Design Document |
| SPMP | Software Project Management Plan |
| SRS | Software Requirements Specification |
| TCF | Technical Complexity Factor |
| TFactor | Technical Factor Value |
| UAW | Unadjusted Actor Weight Total |
| UCP | Use Case Points |
| UUCW | Unadjusted Use Case Weight Total |
| V&V | Verification & Validation |
| WBS | Work Breakdown Structure |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Scope

The purpose of this SPMP is to define the management framework, processes, and methodologies that will be used to ensure the successful completion of the project. This document outlines the necessary project management activities and provides a comprehensive roadmap for planning, execution, monitoring, and control of the project. It serves as a reference for all stakeholders, including team members, academic advisors, and potential sponsors.

The SPMP provides a structured approach to ensure the timely delivery of the software product, adherence to project constraints, and alignment with the project’s objectives and stakeholder expectations.

The scope of this SPMP extends to the following activities and deliverables:

* **WBS:** This document details the hierarchical structure of work packages, sub-deliverables, and tasks required to achieve the project’s objectives. It specifies task dependencies, assignments, and responsibilities for each team member.
* **Project Schedule:** The SPMP outlines a timeline for critical milestones, deliverables, and development increments. It includes key milestones such as the increments of the software.
* **Resource Management:** The SPMP identifies the resources required to complete the project, including human resources, software, hardware, and other essential tools. It outlines the allocation of these resources to specific tasks and deliverables.
* **Effort Estimation:** Multiple estimation techniques are applied, including WBS, Use Case Points (UCP), and Story Points (SP).
* **Verification & Validation Process:** The SPMP defines the V&V strategy for the software product, including testing activities, schedules, and techniques for ensuring product quality and compliance with requirements.
* **Development Environment:** This document outlines the development environment, including programming languages, development frameworks, databases, cloud services, and version control systems used throughout the project.
* **Monitoring and Measurement:** The SPMP defines the approach for tracking the project's progress using process metrics, project metrics, and product metrics. Tools, techniques, and the frequency of data collection are specified to ensure project transparency and control.
* **Discussion and Issues:** Challenges, constraints, and limitations encountered during the development of the SPMP are documented, along with proposed solutions. This section highlights legal, ethical, health, safety, and sustainability considerations related to the software development process.

# Project Schedule

This section specifies the deliverables and work packages for each deliverable. We defined the relationships among them and stated the resource requirements. Also, we provided allocation of budget and resources to work packages.

## Work Packages & Dependencies

In this section, we started by creating a **Work Breakdown Structure (WBS)**. It is an extended and detailed version of the WBS from the Initial Plan. This version is organized into four levels:

**Level 1:** Project

**Level 2:** Deliverables

**Level 3:** Sub-deliverables

**Level 4:** Work packages

**Here is the link to the** [**WBS table**](https://docs.google.com/spreadsheets/d/17LwCG7JbRDwThy15HduDfPUnR2GTTr9M/edit?usp=sharing&ouid=109554406705258888258&rtpof=true&sd=true)**:** [**https://docs.google.com/spreadsheets/d/17LwCG7JbRDwThy15HduDfPUnR2GTTr9M/edit?usp=sharing&ouid=109554406705258888258&rtpof=true**](https://docs.google.com/spreadsheets/d/17LwCG7JbRDwThy15HduDfPUnR2GTTr9M/edit?usp=sharing&ouid=109554406705258888258&rtpof=true&sd=true)

For the documentation deliverables, each main heading of the documents into sub-deliverables, with each document categorized as deliverable. After completing the first increment during the first semester, we planned to create new increments every four weeks. These increments are based on the milestones and deliverables from our initial plan. Increments are listed as deliverables in the WBS.

The WBS includes tasks like **AI implementation**, **mobile application development**, and **backend service creation**, which are spread across increments. We also identified dependencies between work packages. To avoid repetition, if a work package was already dependent on another, we didn't list the same dependency again.

Each task is clearly assigned to specific team members, with previous contributions noted and upcoming responsibilities assigned. This ensures that roles and responsibilities for all deliverables are transparent and organized.

These milestones include Use Cases and tasks aligned with our Use Case diagram drawn in Software Requirements Specification (SRS) document referenced in Section 3.2. The detailed WBS and its relationships ensure that all project activities are planned effectively and assigned clearly to meet our project objectives. Here is the list of the Use Cases that are going to be completed in each increment:

**Increment 1 (Deliverable 9):**

* **UC1 (Login), UC2 (Recover Password), UC3 (Register New User):** Implemented via login, sign-up, and forgot password pages and back-end user service setup.
* **UC4 (Navigate Pages):** Initial mobile app project and bottom navigation.
* **UC7 (Manage Posts), UC8 (View Posts):** Post sharing and feed pages, plus social media service functionality.
* **UC9 (View Profile), UC10 (Edit Profile):** Profile page and user service Create, Read, Update & Delete. (CRUD).
* **UC21 (View AI Chat Lists), UC22 (Manage AI Chats):** Chat screens, chat list, and basic chats service.
* **UC5 (Get Meal Suggestions), UC6 (Get Recipe Instructions):** Initial AI dataset and RAG process to support meal suggestions and instructions.

**Increment 2 (Deliverable 10):**

* **UC15 (Send Notification), UC16 (View Notifications), UC17 (Manage Notifications):** Notification functionality, push notifications service, Kafka integration, and Firebase configurations.
* **UC20 (Search Users):** User search functionality in the back end and mobile app.

**Increment 3 (Deliverable 11):**

* **UC11 (View Achievements), UC12 (Earn Achievement):** Achievements service creation and integration.
* **UC13 (Manage Friends), UC14 (View Friends):** Friend requests and corresponding back-end logic integrated.

**Increment 4 (Deliverable 12):**

* **UC6 (Get Recipe Instructions) Enhancement:** Cooking instructions with step cards, timers, clarification, and sharing the completed dish.
* **UC18 (Share Experience Feedback):** Initial steps toward feedback integrated with recipes and instructions.
* **UC19 (Interact with Posts):** Like/unlike functionality and sharing posts after completing meals.

**Increment 5 (Deliverable 13):**

* **UC19 (Interact with Posts) Enhancement:** Edit and delete posts, and CRUD for comments.

**Increment 6 (Deliverable 14):**

* **UC18 (Share Experience Feedback) Enhancement:** Dedicated feedback service for chat experience and recipe quality feedback.
* **UC19 (Interact with Posts) Enhancement:** Add/edit/delete/view comments functionality completes post interaction features.

## Resource Requirements

1. **Human Resources**

**Project Team:** The number and roles of team members, team members' working time and efforts during the development process of the project. Each team member is responsible for specific areas such as front-end development, backend integration, AI integration, testing, or documentation.  The hourly wages of the project members during the application development and documentation preparation process were determined as **220 TL/hour.**

1. **Hardware Resources**

**Testing and Development Devices:** Team members use personal computers (PCs and Macs) for development tasks. These devices must support environments required for the foodo project. Its cost is **0 TL**(personal devices).

**Electricity Usage:** Continuous use of these devices for development leads to electricity consumption. **~0.03 TL per device per hour**.

**Device Aging and Wear:** Continuous development tasks put stress on personal devices, accelerating hardware wear and tear. This includes potential impacts on battery health, hard drive usage and cooling systems due to prolonged operation. Depreciation Cost is **~1 TL per device per hour.**

1. **Software Resources**
   1. **Development Tools**

* **IDEs: Free** licenses have been used for Visual Studio Code and JetBrains IntelliJ IDEA.
* **GitHub Copilot:** It is used as coding assistant integrated with our IDEs. Its cost is **$20 per month.**
  1. **Software**
* **Qdrant: Free**, self-hosted vector database.
* **Docker:** To containerize the foodo services, Docker is used under **free** license.
* **Kafka:** A message broker to distribute messages between services, Self hosted and will be used under **free** license.

1. **Cloud Resources**
2. **AWS Lambda:** Serverless hosting for AI requests and notifications, pay-as-you-go pricing **($0.20 per 1M requests).**
3. **Amazon S3:** Storage solution for user-uploaded content, pay-as-you-go **($0.023 per GB for storage, $0.005 per 1000 requests).**
4. **Amazon Elastic Container Registry (ERC):**It is an object storage service offering industry-leading scalability, data availability,security, and performance. It is cost is pay-as-you-go **($0.10 per GB for storage, $0.09 per GB for data stream).**
5. **Amazon EC2:** It is avirtual server for cloud computing, selected configuration t3.medium. Estimated to be efficient for our needs, with a pricing of **$0.0416 per hour.**
6. **SaaS Resources**
   1. **OpenAI API:** NLP-powered meal suggestions and chatbot interaction, pay-as-you-go **($2.50 / 1M input tokens and $10 /1M output tokens).**
   2. **Google Vertex AI:** ML model hosting and training for recipe recommendations, pay-as-you-go **($0.0003125 / 1k characters).**
   3. **Claude:** It delivers advanced NLP functionalities. Its cost is **$0.80 / MTok for Input, $4 / MTok** for output.
7. **Support Software**
   1. **ChatGPT Monthly Subscription:** For all team members, ChatGPT Plus will cost **$24 per month.**
   2. **MS Office Subscription Billing:** Team member Cemal Fırat Dağ purchased the MS Office family package for **309.99 TL per month.**
8. **Transportation and Communication Resources**

The fuel price will be calculated between **~45 TL per liter.**

**Alper Çelik:** He uses his own private vehicle and consumes 10.5 liters of fuel per 100 km in the city.

**Batuhan Duras:** He uses his own private vehicle and consumes 8.3 liters of fuel per 100 km in the city.

**Cemal Fırat Dağ:** He uses public transportation and pays **10.5 TL** for every hop on.

**Baturalp Sönmez:**  He uses both public transportation and electric bikes and pays **10.5 TL** for every hop on. Using his electric bike, he has an expense of **~3 TL** for every 100 km.

**Davut Durmaz:** He uses public transportation and pays **10.5 TL** for every hop on.

1. **Internet Consumption**

Reliable internet is required for accessing online resources, cloud services and collaborative tools like GitHub and Slack. The cheapest subscription has a monthly fee of **469.90 TL** from the internet operator TurkNet with a 12-month advance payment discount.

1. **Communication**
   1. **Zoom:** It is used for formal meetings, stakeholder discussions, and advisor feedback sessions. It is used under **free** license.
   2. **Slack:** It is used instead of Discord due to restrictions in Turkey. **Free** version is used.
2. **Facilities**
   1. **Office:** The office serves as the primary working environment for the foodo team. It provides a centralized space for team collaboration, meetings, and development activities. The cost is **~3500 TL per month**.
3. **Consumables**

Consumables play a crucial role in our team’s energy and morale during the development of the foodo.

* 1. **Energy Drinks and Coffee:** To stay focused and productive our team frequently consumes energy drinks like Monster and Red Bull, along with coffee. The cost is **~80 TL** per cup of coffee, **~50 TL** for every can of an energy drink.
  2. **Snacks:** A selection of snacks are regularly consumed during our team sessions, such as Eti Pop Kek and Doritos Hot Shots. The cost is **~30 TL** per snack.
  3. **Vending Machine Beverages:** Quick and convenient options like vending machine coffee and bottled water are often utilized during on-campus meetings or office work. The cost is **~15 TL** per item on vending machine.
  4. **Team Meals:** During extended work sessions or milestone celebrations, our team occasionally orders meals or shares traditional Turkish foods like pide, doner or lahmacun. The cost is **~300TL** per meal.

1. **Domain Cost**

We bought [www.foodo.cfd](https://foodo.cfd/) from namecheap and it costed us **$1.67 for a year.A screenshot of a computer

Description automatically generated**

Figure 1. foodo.cfd Order Summary

## Cost Estimation, Net Present Value, Budget and Resource Allocation

As a team, we carefully documented and analyzed our incurred expenses and utilized these figures to project further costs. This process was grounded in the calculations and estimations derived from the costs outlined in our **Resource Requirements.**

Based on our calculations, the cost of one hour of work per individual amounts to **221.03 TL**, a figure that includes device depreciation to account for equipment wear and tear over time, employee wages covering all personnel costs, and electricity consumption reflecting the energy usage during working hours. Based on our WBS, we distributed need effort per month as listed below:

Table 1. Employee Cost Estimation



The detailed analysis of effort allocation helped us determine precise labor costs across project phases.

To support efficient project development and documentation, we have estimated various operational expenses that are necessary for the smooth execution of our work. These include services, subscriptions, workplace rent, consumables, and transportation costs. The costs associated with these items were calculated carefully, including currency conversion for relevant services, and were based on anticipated usage and past experiences to ensure an accurate and realistic budget projection:

Table 2. Development Expenses



Our project requires investments in cloud services and APIs to support hosting, data storage, and application functionality, with costs estimated based on prior experience and knowledge of similar projects. This includes the domain name **foodo.cfd**, purchased in October 2024 for an annual cost of **58.02 TL**, as well as expenses for cloud services covering hosting, storage, and related infrastructure, and API usage costs calculated based on anticipated call volumes and associated fees.

Table 3. Cloud and API Costs



After identifying and categorizing all projected expenses, we developed a **cash flow diagram** (**Figure 2. Cash Flow Diagram**). This diagram outlines monthly outflows, providing a clear financial trajectory for the project. It allows us to visualize and manage resources effectively over time.

A diagram of a timeline

Description automatically generated

Figure 2. Cash Flow Diagram

To evaluate the overall financial viability of our project, we performed **NPV calculations** for each month’s cash flow (**Table 4. NPV Calculation Table**). By summing up the NPVs, we derived the project’s total cost, amounting to **504,171.2042 TL**.

This comprehensive analysis ensures that our budgeting aligns with the project’s needs and mitigates potential financial risks.

Table 4. NPV Calculation Table



The structured approach to expense estimation and financial planning has provided us with:

**Clarity**: A well-defined view of our costs and resource allocation.

**Accuracy**: Precise estimations that account for both direct and indirect expenses.

**Viability**: An understanding of the project’s financial feasibility through NPV analysis.

These efforts will enable us to manage resources effectively, ensure project success, and achieve our goals.

# Project Effort Estimation

## Work Breakdown Structure (WBS) - Decomposition-Based Estimation

All the estimated and calculated efforts given here were referenced in Section 2.1 Work Packages & Dependencies. As we have completed some Work Packages in our WBS, we used this historical data to estimate future Work Packages in our Supporting Software Engineering Processes. Also, for the Software Product Processes we estimated based on our work experience.

**Supporting Software Engineering Processes:** Our Supporting Software Engineering Processes include the following items as listed below:

* **Deliverable 1. Initial Plan v1:** Already done in **52 hours.**
* **Deliverable 2. Initial Plan v2:** Already done in **35 hours.**
* **Deliverable 3. SRS v1:** Already done in **60 hours**.
* **Deliverable 4. SRS v2:** Already done in **42 hours.**
* **Deliverable 5. SPMP v1:** Already done in **48 hours.**
* **Deliverable 6. SPMP v2:** Estimated to finish in **34 hours.**
* **Deliverable 7. SDD v1:** Estimated to finish in **50 hours.**
* **Deliverable 8. SDD v2:** Estimated to finish in **35 hours.**
* **Software Product Processes:** Our Software Product Processes include the following items as listed below:

**Deliverable 9. First Increment:** Estimated to finish in **365 hours** (20 hours estimated for presentation).

**Deliverable 10. Second Increment:** Estimated to finish in **345 hours.**

**Deliverable 11. Third Increment:** Estimated to finish in **345 hours.**

**Deliverable 12. Fourth Increment:** Estimated to finish in **345 hours.**

**Deliverable 13. Fifth Increment:** Estimated to finish in **345 hours.**

**Deliverable 14. Sixth Increment:** Estimated to finish in **345 hours.**

* **Deliverable 15. Final Presentation:** Estimated to finish in **40 hours.**

In total our Supporting Software Engineering Processes cost us **237 hours** and we are planning to spend **119 hours** on these Work Packages. Also, we estimated that our software product processes are going to cost us **2090 hours** and we also estimated to spend **40 hours** for the final presentation. In total we estimated the effort which is needed for our project as **2486 hours** in **total**.

## Use Case Based Estimation

A diagram of a diagram

Description automatically generated

Figure 3. UML Use Case Diagram [1]

Initially, we calculated the unadjusted actor weights. We classified **A2 (Clerk)**, **A3 (LLM Provider)**, and **A4 (Google Firebase Cloud Messaging)** as *Simple Actors*, and **A1 (User)** as a *Complex Actor*. Using this classification, we computed the **Unadjusted Actor Weight Total (UAW)**.

Next, we classified our use cases as follows:

* **Simple Use Cases**: UC1, UC2, UC4, UC8, UC9, UC11, UC14, UC15, UC16, UC17, UC18, UC20, UC21
* **Average Use Cases**: UC3, UC7, UC10, UC13, UC19, UC22
* **Complex Use Cases**: UC5, UC6, UC12

We identified 13 *Simple*, 6 *Average*, and 3 *Complex* use cases, and used this information to calculate the **Unadjusted Use Case Weight Total (UUCW)**.

Afterwards, we assigned values between 0 and 5 to technical factors, which allowed us to compute the **Technical Factor Value (TFactor)**. We then calculated the **Technical Complexity Factor (TCF)** using the following formula:

Similarly, we assigned values between 0 and 5 to environmental factors, which were used to calculate the **Environmental Factor Value (EFactor)**. The **Environmental Factor (EF)** was then computed using the formula:

Table 5. Unadjusted Actor Weights

A table with text on it

Description automatically generated

Table 6. Unadjusted Use Case Weights

A blue and white table with numbers

Description automatically generated

Table 7. Technical Complexity Factors

A blue screen with black text

Description automatically generated

Table 8. Environmental Factors

A blue box with black text

Description automatically generated

Next, we calculated the **Adjusted UCP** using the formula:

Substituting the values:

Since no **EFactors** (E1 through E6) were assigned a value less than 3, and only **E7** had value greater than 3 (with a count of 1), we selected a **Person Hour Multiplier (PHM)** of 20. Finally, we calculated the **Effort** (in person-hours) using the formula:

Substituting the values:

## Agile Estimation

Initially, we chose the T-Shirt Sizing scale for our story point estimation because our GitHub organization had already set up a project using this scale, and we wanted to maintain consistency. Following this, we created a Story Point Estimation Matrix.

Table 9. Story Point Estimation Matrix



We first created user stories based on our use cases, then we played planning poker to assign SP to our user stories. After the poker session, we decided on the T-Shirt Sizing scale to ensure consistency with the project setup in our GitHub organization. As a group, we then assigned SP based on the results of our poker game. The final story point assignments were recorded in the **User Story Points Table**.

Table 10. User Story Points



As a result, we calculated the total SP to be 236. However, it’s important to note that our data is limited, which means these estimates have a higher likelihood of being inaccurate. From the available data, we know that our team completed 4 SP in 8 hours. This suggests that we require approximately 2 hours to complete each story point, which amounts to a total of 472 hours. **However, this estimation should be viewed with caution**. As more data becomes available and our team gains more experience, these numbers are likely to change. We anticipate that our understanding of time per story point will evolve, potentially leading to a revision of the total hours required.

## Discussion

We computed our estimations using three different approaches, each yielding different results due to variations in scope and methodology. The **WBS Decomposition-Based Estimation** resulted in *X* hours, the **Use Case-Based Estimation** yielded 2347.02 person-hours, and the **Agile Estimation** came to 472 hours.

The discrepancies arise because these methods focus on different aspects of the project. For example, **WBS Decomposition-Based Estimation** relies on detailed task decomposition, typically producing more granular and potentially accurate estimates if the scope is well-defined. **Use Case-Based Estimation** systematically applies weighted factors to actors, use cases, and environmental conditions. **Agile Estimation,** which is based on SP and velocity, provides a dynamic approach but depends heavily on team experience and historical data, making it less reliable early in a project.

While we calculated 472 hours using Agile Estimation, this was based on completing only four SP, which provides insufficient data for accurate estimation at this early stage. We anticipate that Agile Estimation will become more reliable as the project progresses and more data is gathered. Use Case-Based Estimation, which resulted in 2347.02 person-hours, offers a more structured and detailed approach, making it currently more realistic for estimating software development effort. Similarly, the Work Breakdown Structure (WBS) method, which calculated 2484 hours provides detailed task-level granularity and is well-suited for projects with clearly defined scopes, contributing to its reliability at this stage.

At this point, we consider the **WBS Decomposition-Based Estimation** and **Use Case-Based Estimation** the most accurate representation of the project.[[2]](#footnote-2)

# Project Monitoring and Measuring

1. **Monitor Progress**

For monitoring our progress as a team, a combination of weekly meetings and tools such as Slack, Zoom and GitHub is utilized for effective communication and task tracking.

1. **Weekly Meetings:** During the first semester, we held structured weekly meetings to review the progress of deliverables, such as the SRS, SPMP and SDD. These meetings are logged on to our GitHub inside our project-docs, inside meeting.md. Also, meeting notes are documented and shared in Slack channels. These meetings allow us to discuss completed tasks, evaluate progress against milestones and address any potential delays.

In the second semester, we plan to conduct two meetings per week:

* **Planning Meetings (on Mondays):** The project goals will be set weekly, and tasks will be prioritized from the product backlog which are set on GitHub. Additionally, dependencies will be identified in these meetings. These meetings will ensure the team starts the week with a clear direction.
* **Review Meetings (on Fridays):** The challenges that occurred and the task progress will be discussed in these meetings. Also, progress will be assessed, feedback will be provided and what needs to be improved or adjusted will be discussed.

1. **Stakeholder Engagement:** Formal Zoom meetings are scheduled bi-weekly with stakeholders to present progress, address challenges and collect feedback. To exemplify, we have met our advisor Dr. Serkan Genç to show our progress and we have taken needed feedback from Dr. Oumout Chouseinoglou after the document assignments. Meeting notes and action items from these discussions are stored in Slack for easy access by the team. Stakeholders also track the project’s development via GitHub, where milestones and tasks are visible.
2. **Task Tracking with GitHub**

GitHub serves as the primary tool for task management, progress tracking and documentation. Issues, milestones and pull requests are updated regularly to provide real-time insights into development progress at our main repository foodoHub, which includes the project repositories for both the mobile part and AI part and project docs.

1. **Communication Channels**

Internal communication is facilitated through Slack, which hosts channels for task management, resources and meeting notes. Urgent updates or informal discussions are managed through WhatsApp groups for both formal and informal communication, and announcements. Also, GitHub provides a centralized platform for tracking technical tasks as mentioned above. [[3]](#footnote-3)

1. **Metrics Collection**
2. **Project Metrics**

**Task Completion Rate Metric:** It tracks the percentage of completed tasks out of the total tasks in the sprint backlog. It is measured weekly by analyzing the progress of tasks on GitHub Issues. The data is tracked in GitHub Projects, providing visual insights via its built-in dashboards (such as prioritized backlog and status board table). This metric ensures that the team stays aligned with plans and allows for quick identification of delays or bottlenecks.

A screenshot of a computer

Description automatically generated

Figure 4. GitHub Prioritized Backlog Page of foodo

**Feature Completion Coverage Metric:** It tracks the percentage of Work Packages defined under WBS that will be fully implemented. Each work package will be linked to GitHub milestones and their respective Issues. This metric will be updated as Issues are closed, and the corresponding features are validated. It ensures that all planned features, such as AI meal suggestions or social posting functionalities, are developed and meet the project’s goals. By monitoring feature completion on the review meeting, our team will assess progress and address any delays in implementing critical requirements. [[4]](#footnote-4)

1. **Product Metrics**

**API Response Time for Share Post Endpoint:** Measures the average time taken for the "Share Post" API endpoint to respond to a user request. This metric will be logged during testing phases or app updates and stored in PostgreSQL. Ensuring optimal response times is critical to maintaining usability.

**Achievement Completion Distribution Metric:** It tracks the percentage of users unlocking each achievement badge, such as the "Kebabster" badge for cooking kebab multiple times. This metric provides insights into the popularity of milestones like "Recipe Mastery" or "Social Sharing." Data will be collected in real-time and will be stored in PostgreSQL, categorized by achievement type. For instance, the system may track that 40% of users unlocked the "Kebabster" badge at Level 1 and 20% at Level 2. This analysis helps the team optimize the gamification system, focusing on user preferences and enhancing engagement across features.

1. **Process Metrics**

**Cycle Time Metric:** It calculates the average time taken to complete a task from "To Do (In progress)" to "Done." This metric will be measured weekly using GitHub Projects and will be stored within GitHub workflows. It identifies bottlenecks and supports process improvements.

A screenshot of a computer

Description automatically generated

Figure 5. GitHub Status Board of foodo

**Team Velocity Metric:** It measures the average number of SP completed per sprint, calculated during retrospectives. Velocity data will be documented in sprint reports to evaluate productivity and forecast future capacity. [[5]](#footnote-5)

# Product Verification and Validation

The Product Verification and Validation Plan outlines the strategies, tools, and schedule we will use to ensure the quality and correctness of "foodo". The plan aims to verify that the product is built according to the specified requirements (verification) and validates that it meets the stakeholders’ needs (validation). The following sections detail our approach, tools, schedule, and sample test cases.

1. **Techniques for V&V**
2. **Verification Techniques**

This section outlines the methods used to verify the system's commitment to defined requirements and coding standards to ensure functional correctness and code quality.

1. **Code Reviews**

* Team members will perform systematic peer reviews of code changes to detect bugs and ensure commitment to coding standards.
* Focus on identifying errors early in the development cycle.

1. **Static Code Analysis**

* Use automated tools to analyze the code for errors, vulnerabilities, and deviations from coding standards.

1. **Unit Testing**

* Test individual components or modules to ensure they function as expected in isolation.

1. **API Testing**

* Validate the backend API endpoints to ensure they handle requests and responses correctly.

1. **Validation Techniques**

This section focuses on activities designed to validate that the application meets user needs and performs well under realistic usage scenarios.

1. **User Acceptance Testing (UAT)**

* Involve real users and stakeholders to validate the app’s features and usability.

1. **Compatibility Testing**

* Test the app on various devices (iOS and Android) to ensure consistent performance.

1. **Exploratory Testing**

* Simulate real-world usage scenarios to uncover potential issues not covered by predefined test cases.

1. **Fault Injection Testing**

* Introduce errors intentionally to observe how the system handles unexpected conditions.

1. **Usability Testing**

* Evaluate the user interface and navigation to ensure consistent user experience.

1. **V&V Tools**
2. **Code Reviews**

* **IntelliJ IDEA**: To review and refactor code with advanced navigation, syntax highlighting, and integrated version control.
* **GitHub Pull Requests**: For peer code reviews, inline comments, and change tracking.

1. **Static Code Analysis**

* **ESLint:** To statically analyze code to quickly find problems. ESLint will validate formatting, naming conventions, and best practices for JavaScript/TypeScript, ensuring readability and maintainability
* **SonarLint**: To perform on-the-fly analysis in the IDE and catch bugs or code smells early.
* **Prettier**: For enforcing coding style guidelines and ensuring consistent formatting.

1. **Unit Testing Frameworks**

* **JUnit**: For backend services, providing extensive support for testing Java applications.

1. **API Testing Tools**

* **Postman**: For manual testing of API endpoints, validating request-response pairs, and simulating edge cases.

1. **Device Testing**

* **Expo Go**: For live testing and debugging of React Native applications during development. Aimed to test both IOS and Android.

1. **User Feedback Tools**

* **Google Forms**: For gathering structured feedback from beta testers and users during UAT.
* **In-App Survey Integration**: Custom feedback forms embedded within the application for quick user reviews.

1. **Issue Tracking**

* **GitHub Issues**: For logging, categorizing, and tracking bugs, feature requests, and enhancement tasks.
* **GitHub Projects**: To have an adaptable spreadsheet, task-board, and road map that integrates with our issues.

1. **Continuous Integration/Delivery**

* **GitHub Actions**: To automate build, test, and deploy pipelines, ensuring high-quality code delivery.

1. **Usability and Accessibility Testing**

* **Material Design Guidelines**: We will adhere to the Material Design Guidelines to ensure consistency and usability across Android and iOS platforms. Using Material Design's features, we will create an accessible user experience. Material 3 will be the primary framework, providing support for modern design principles, including dark themes and improved legibility.
* **Prototype Testing**: Early-stage UI/UX designs will be tested with stakeholders and users to gather.

1. **Logging and Monitoring**

* **AWS CloudWatch**: For monitoring application logs and system performance in cloud environments.

1. **V&V Schedule**

The V&V process will follow a phased approach aligned with the development timeline. The schedule ensures that all quality assurance tasks are completed before deployment.

Table 11. Verification and Validation Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Schedule | Activities | Tools |
| Requirement Verification | 16.09.2024 – 04.12.2024 | - Review SRS. - Ensure requirements align with functional and non-functional goals. | -GitHub Issues  -Manual Review |
| Design Verification | 08.12.2024 – 09.01.2024 | - Validate system architecture diagrams and database schema against requirements. | -IntelliJ IDEA |
| Static Code Analysis | 03.12.2024 – 09.06.2025 | - To identify vulnerabilities and code smells. | -SonarLint  -ESLint  -Prettier |
| Unit Testing | 07.01.2025 – 09.06.2025 | - Develop and execute unit tests for isolated components. - Focus on critical modules first. | -JUnit |
| User Acceptance Testing | 05.01.2025 – 09.06.2025 | - Engage beta testers for usability feedback. - Validate application against user expectations. | -Expo Go  -Jotform |
| Exploratory & Fault Testing | 06.12.2024 – 01.06.2025 | - Simulate real-world usage scenarios. - Test edge-case behaviors. | -GitHub Issues |
| Usability & Accessibility Testing | 06.12.2024 – 01.06.2025 | - Evaluate UI/UX designs for consistency and accessibility. - Test responsiveness on iOS/Android. | -Prototype Testing  -Accessibility Checklists |
| Deployment Readiness Validation | Before each increment:  - 08.01.2025  - 26.01.2025  - 23.02.2025  - 23.03.2025  - 20.04.2025  - 18.05.2025  - 08.06.2025 | - Final validation of deployment readiness. - Confirm integration with cloud services. | -GitHub Actions  -AWS CloudWatch |

1. **Sample Test Cases**

To ensure comprehensive V&V of the "foodo", we prepared sample test cases for each planned activity. These test cases include all necessary details, from prerequisites to expected results, providing a clear framework for effective testing.

1. **Verification Activities and Sample Test Cases**

**Code Reviews**

Table 12. Code Review Test Case

|  |  |
| --- | --- |
| **Name** | **Code Review for Login Module** |
| **Definition** | Ensure that the login module code sticks to coding standards and is well-documented. |
| **Prerequisite Conditions** | The login module code is written and pushed to the repository. |
| **Test Inputs** | Login module code file. |
| **Assumptions & Constraints** | The reviewer has access to the repository and follows Prettier style guidelines. |
| **Test Steps** | **Action** |
|  | 1. Open the login module code in IntelliJ IDEA. |
| 2. Review the file for proper formatting and comments. |
| 3. Check if all functions and methods are documented. |

**Static Code Analysis**

Table 13. Static Code Analysis Test Case

|  |  |
| --- | --- |
| **Name** | **Static Analysis for Code Vulnerabilities** |
| **Definition** | Identify code smells and vulnerabilities in the codebase using SonarLint. |
| **Prerequisite Conditions** | Codebase is complete and open in IntelliJ IDEA. |
| **Test Inputs** | Codebase files. |
| **Assumptions & Constraints** | SonarLint is installed and functional in IntelliJ IDEA. |
| **Test Steps** | **Action** |
|  | 1. Open the project in IntelliJ IDEA. |
|  | 2. Run SonarLint analysis on the entire project. |
|  | 3. Review the report for high-priority issues. |

**Unit Testing**

Table 14. Unit Testing Test Case

|  |  |
| --- | --- |
| **Name** | **Testing Login Functionality** |
| **Definition** | Verify that the login function works as expected in isolation. |
| **Prerequisite Conditions** | The login function is implemented. |
| **Test Inputs** | Valid and invalid login credentials. |
| **Assumptions & Constraints** | JUnit framework is set up and configured. |
| **Test Steps** | **Action** |
|  | 1. Input valid credentials and call the login function. |
|  | 2. Input invalid credentials and call the login function. |
|  | 3. Input empty fields and call the login function. |

**API Testing**

Table 15. API Testing Test Case

|  |  |
| --- | --- |
| **Name** | **Testing User Login API Endpoint** |
| **Definition** | Validate that the login API endpoint handles requests correctly. |
| **Prerequisite Conditions** | Backend API is deployed. |
| **Test Inputs** | POST request with valid and invalid JSON payloads. |
| **Assumptions & Constraints** | The API is accessible via Postman. |
| **Test Steps** | **Action** |
|  | 1. Send a POST request with valid credentials. |
|  | 2. Send a POST request with invalid credentials. |
|  | 3. Send a POST request with empty payload. |

1. **Validation Activities and Sample Test Cases**

**User Acceptance Testing (UAT)**

Table 16. UAT Test Case

|  |  |
| --- | --- |
| **Name** | **UAT for Meal Suggestion Flow** |
| **Definition** | Validate user satisfaction meal suggestion flow in the app. |
| **Prerequisite Conditions** | The app is deployed on iOS and Android devices for testing. |
| **Test Inputs** | Test accounts, device access, and the deployed app. |
| **Assumptions & Constraints** | Stakeholders have basic knowledge of using mobile applications. |
| **Test Steps** | **Action** |
|  | 1. Open the app and log in with valid credentials. |
|  | 2. Start communicating with the chatbot. |
|  | 3. Ask for meal suggestions. |

**Compatibility Testing**

Table 17. Compatibility Testing Test Case

|  |  |
| --- | --- |
| **Name** | **Cross-Device Compatibility** |
| **Definition** | Test the app functionality on different devices (iOS and Android). |
| **Prerequisite Conditions** | The app is installed on multiple devices with different OS versions. |
| **Test Inputs** | Devices (e.g., iPhone 13, Xiaomi Mi11T) app installation files. |
| **Assumptions & Constraints** | Devices have stable internet connections and compatible OS versions. |
| **Test Steps** | **Action** |
|  | 1. Launch the app on each device. |
|  | 2. Navigate through the app and perform basic tasks (e.g., login, check feed). |
|  | 3. Ask for a recipe from the chatbot and confirm. |

**Fault Injection Testing**

Table 18. Fault Injection Test Case

|  |  |
| --- | --- |
| **Name** | **Error Handling During Chatbot Interaction** |
| **Definition** | Validate how the app handles errors during an interaction with the chatbot. (e.g., network issues). |
| **Prerequisite Conditions** | The app includes fully implemented chatbot UI and error handling logic. |
| **Test Inputs** | Test chatbot interaction, simulated network issues. |
| **Assumptions & Constraints** | Test environment can simulate network issues. |
| **Test Steps** | **Action** |
|  | 1. Start the chatbot interaction process and send messages. |
|  | 2. Simulate a network failure mid-payment. |
|  | 3. Retry chatting after restoring the connection. |

**Usability Testing**

Table 19. Usability Testing Test Case

|  |  |
| --- | --- |
| **Name** | **Testing User Interface Accessibility** |
| **Definition** | Ensure the app's user interface is accessible and easy to use. |
| **Prerequisite Conditions** | UI components are designed and integrated. |
| **Test Inputs** | Keyboard and touchscreen navigation. |
| **Assumptions & Constraints** | Input tools are available for testing. |
| **Test Steps** | **Action** |
|  | 1. Use touchpad to navigate through the app. |
|  | 2. Test navigation using keyboard shortcuts. |
|  | 3. Check color contrast in UI components. |

This plan ensures that foodo meets both functional and non-functional requirements while exceeding user expectations. The outlined V&V activities, coupled with test cases and a structured schedule, provide a rich framework for delivering a high-quality product.

# Software Development Environment

The foodo is being developed using a modern microservices architecture to ensure scalability, flexibility, and maintainability. Our approach divides the application into independent, modular services, each focused on specific functionalities like AI-powered cooking assistance and social features. This architecture allows for seamless integration of new features, efficient scaling, and rapid updates based on user feedback. Here is the table of details of foodo’s Software Development Environment.[[6]](#footnote-6)

Table 20. Details of Software Development Environment



# Discussions

## Limitations and Constraints

While preparing the SPMP, our team faced several limitations and constraints:

* **Time Constraints**: Balancing personal, academic, and professional commitments made it challenging to allocate sufficient time for SPMP preparation. Since all of our team members are working part-time jobs, It was hard for us to meet. The schedule of each team member was constraining our meeting times. But by looking at our priorities we managed to meet all of us in person multiple times for the initial plan.
* **Work Distribution Challenges:** Disagreements came up regarding task assignments and workload distribution within the team. Differing availability and priorities among team members worsened these challenges. At times, these disagreements escalated into verbal conflicts, with elevated emotions and arguments about sensed injustices in the effort contributed. Although these situations caused temporary disruptions in the workflow, the team proactively addressed the issues through scheduled meetings and open discussions. By allowing each member to voice their concerns and collaboratively revisiting task assignments, we successfully restored harmony and ensured a fair distribution of responsibilities.[[7]](#footnote-7)
* **Communication Challenges:** During the initial phases, the team used Discord for communication. However, due to network restrictions, Discord became inaccessible. This forced us to switch to Slack for team communication, which initially disrupted our workflow. Nevertheless, the team adapted quickly, and Slack proved effective in maintaining smooth communication.

## Health and Safety Issues

The intense workload during the SPMP preparation led to the following health and safety issues:

* **Sleep Problems**: Many team members experienced reduced sleep hours while trying to balance their academic responsibilities and SPMP preparation.
* **Poor Eating Habits**: Unhealthy eating patterns, such as skipping meals or consuming excessive fast food and caffeine, were prevalent during this period.
* **Physical Discomfort**: Prolonged screen time caused neck, back, and muscle pain.
* **Family Emergencies**: Batuhan had to leave a night of SPMP work early due to family matters, which affected task progress temporarily.
* **Professional Impact**: Batuhan arrived late to work one day because of staying up late to finalize parts of the SPMP. Cemal missed an important business meeting because he slept very late after working on the SPMP.
* **Energy Drink Access Issues**: As selling energy drinks is banned in campus, Batuhan and Baturalp had to travel to Bilkent Center Migros to buy Monsters to bear the SPMP workload. This journey caused a loss of gas, money, time, and general joy, further adding to the overall stress of the preparation process.
* A receipt on a table

  Description automatically generated

Figure 6. Receipt for Energy Drinks

## Legal Issues

No legal issues were encountered during the preparation of the SPMP.

## Economic Issues and Constraints

Team members incurred personal expenses for tools and resources used during preparation, as there was no official project budget. For example:

* Costs for transportation to attend in-person meetings.
* Usage of personal equipment such as laptops and mobile devices.
* Additional expenses were incurred due to the need to purchase energy drinks from Bilkent Center Migros, including gas and product costs.

## Sustainability

To promote sustainability during the SPMP preparation:

* **Digital Collaboration Tools:** We minimized paper usage by utilizing digital tools like Google Docs and Microsoft Office for document editing and task management.
* **Efficient Meetings:** Where possible, we opted for virtual meetings to reduce transportation-related carbon footprints.
* **Campus Walking:** For on-campus meetings, team members choose to walk instead of using personal vehicles.

## Ethical Issues

Two of our team members, Batuhan and Cemal used company-provided laptops for academic purposes, raising minor ethical concerns about using employer-owned resources for non-work-related activities. However, this was necessary to maintain project progress.

## Multidisciplinary Collaboration

There were no multidisciplinary collaborations during the preparation of the SPMP. The work was carried out entirely within the team's expertise in software project management and development.

# References

|  |  |
| --- | --- |
| [1] | A. Çelik, C. F. Dağ, B. Duras, D. Durmaz and B. Sönmez, "Software Requirements Specification v2 of Team 11," Ankara, 2024. |

1. GenAI tool: ChatGPT 4o

   Prompt: “Can you write us an Executive Summary based on the provided and previous documents.”

   Rationale: To complete Executive Summary Part [↑](#footnote-ref-1)
2. GenAI tool: ChatGPT 4o

   Prompt: “Can you improve the language of the given text”

   Rationale: To improve language. [↑](#footnote-ref-2)
3. GenAI tool: ChatGPT 4o

   Prompt: “can you explain the communication channels that I wrote clearer and shortly”

   Rationale: To explain the sentences clearly [↑](#footnote-ref-3)
4. GenAI tool: ChatGPT 4o

   Prompt: “what are the most used project metrics related to GitHub’s project features”

   Rationale: To take an idea about project metrics [↑](#footnote-ref-4)
5. GenAI tool: ChatGPT 4o

   Prompt: “give process metrics example”

   Rationale: To think wisely about this topic [↑](#footnote-ref-5)
6. GenAI tool: ChatGPT 4o

   Prompt: “Improve language professionally”

   Rationale: To have a more professional text [↑](#footnote-ref-6)
7. GenAI tool: ChatGPT 4

   Prompt: “Formalize text without changing the context. Censor the names used.”

   Rationale: To make sure that the problem we experienced is clearly addressed. [↑](#footnote-ref-7)