|  |  |
| --- | --- |
| **Module Code** | **CN 7021** |
| **Module Title** | **Advanced Software Development** |
| **LSBF Coursework Instructor** | **Francis Ho** |
| **Coursework Title** | **Dining Explorer Platform** |
| **Weighting** |  |
| **Handout Date** |  |
| **Coursework Submission Date** |  |
| **Students’ Name & Students’ ID** | |  |  | | --- | --- | | **Student Name(s)** | **Student ID Number(s)** | | Timur Mamadaliyev | S1033860 | | Zhang ShanShan | S1034642 | | Zhang Thai | U2720444/S1033859 | | YANG MINGXIN | S1034638 | |
| **Documents uploaded in LSBF Canvas:** | ⭘ Assignment Report  ⭘ Presentation Report  ⭘ Source Code, Scripts & Output  ⭘ Turnitin Report |
| **UEL Turnitin Submission Similarity Index %** |  |
|  | |

**Table Of Content**

[**1. Executive Summary** 3](#_Toc166061038)

[**2. Project Background** 4](#_Toc166061039)

[**3. Content** 8](#_Toc166061040)

[3.1 Product vision, roadmap, prototype design 8](#_Toc166061041)

[3.2 Agile methodology employed (Cockburn, A., 2001) 10](#_Toc166061042)

[3.3 Features, scenarios, (user) stories, and product backlog 13](#_Toc166061043)

[3.4 Software architecture 25](#_Toc166061044)

[3.5 Security and privacy concerns 31](#_Toc166061045)

[3.6 Test plan, test examples, and results 33](#_Toc166061046)

[3.7 DevOps and code management 34](#_Toc166061047)

[**4. Conclusion** 36](#_Toc166061048)

[**5. Work breakdown by each member written for the rest of the members** 37](#_Toc166061049)

[**6. References** 38](#_Toc166061050)

[**Appendix A. Scrum meeting description** 39](#_Toc166061051)

[**Appendix B. Use case Templates** 40](#_Toc166061052)

[**Appendix C. Turnitin report** 43](#_Toc166061053)

# **1. Executive Summary**

The Dining Explorer project aims to revolutionize the dining experience by providing users with a comprehensive platform for restaurant discovery, food ordering, table reservations, and user reviews. The platform targets a diverse audience, including local residents, tourists, and food enthusiasts, seeking convenient and enjoyable dining options.

Users can search for restaurants based on cuisine type, price range, location, and atmosphere, enhancing their dining choices and experiences. Additionally, seamless integration with food delivery services allows users to order their favorite dishes for delivery to their doorstep, ensuring convenience and satisfaction. Real-time table availability checking and reservation management enable users to plan and book dining experiences with an opportunity to leave reviews and ratings.

The target audience for this application are tourists, food enthusiasts, and everyone who wants to find an amazing place to eat.

# **2. Project Background**

The Dining Explorer platform aims to revolutionize the way people discover, choose, and enjoy dining experiences. With a focus on convenience, affordability, and personalization, this platform will serve as a comprehensive tool for individuals seeking to find the perfect dining spot. Whether it's a resident looking for a new favorite restaurant or a tourist exploring a new city, Dining Explorer will provide valuable insights and services to enhance the dining experience.

Historically, each city and country have great gastronomical places, that are aimed at tourists. But spreading the information about it was always a resident’s storytelling or legends from experienced tourists. During the Internet epoch and the development of different applications and websites, it has become far easier to find information about nice eating places. However, mostly it is just some reviews from the users on some strange websites, or ratings without any criteria (just 5 stars doesn’t say much about the meal type or prices). Also, after even finding an interesting name, you have to find it in the city, which also could be quite a challenge (similar names, especially if you are in another speaking country).

Finding a suitable restaurant can be a daunting task, especially in unfamiliar locations. Tourists often struggle to locate restaurants that align with their preferences and budgets, leading to suboptimal dining experiences. Additionally, the process of checking restaurant reviews, making reservations, and coordinating food delivery from afar can be time-consuming and complicated. It is annoying that you have to use sometimes almost 3-4 different services to find what you find.

Prior Solutions:

There are several online platforms and directories (e.g., Yelp, TripAdvisor, Zomato) that offer restaurant discovery, reviews, and ratings.

But they have the following gaps:

i. Lack of personalized recommendations based on user preferences.

ii. Limited integration with food delivery services, requiring users to switch between platforms.

iii. Reservation systems are often separate, leading to a disjointed user experience.

iv. Limited filtering options, especially related to crowdness, atmosphere, and weather conditions.

Project Goals and Objectives:

Empower Users: Provide users with a user-friendly platform to easily discover restaurants based on taste preferences, affordability, and location.

Enhance Tourist Experience: Offer tourists a reliable tool to find culturally authentic, affordable, and well-reviewed dining options in new cities.

Integrated Food Delivery: Enable users to order food directly from the platform, connecting them with delivery services for added convenience.

Streamlined Reservations: Allow users to book tables at restaurants directly through the platform, simplifying the reservation process.

Transparent Reviews: Display comprehensive restaurant reviews, ratings, and user feedback to assist in decision-making.

Advanced Filtering: Offer advanced filters such as crowdness, popularity, atmosphere, and weather conditions for outdoor seating.

Stakeholders:

Users: Individuals seeking dining options, including residents and tourists.

Restaurant Managers: Owners and managers of restaurants featured on the platform.

Food Delivery Services: Partners providing delivery options through the platform.

Features and Functionality:

Restaurant Discovery:

Users can explore a wide range of restaurants based on cuisine, price range, location, and user ratings.

For tourists, a special "Tourist Mode" offers curated recommendations for authentic local dining experiences.

Integrated Food Delivery:

When a user selects a restaurant, they can check if any partnered food delivery services offer delivery to their location.

Seamless integration with popular food delivery platforms allows users to place orders directly from the platform.

Table Reservations:

Users can view real-time availability and book tables at restaurants through the platform.

Confirmation notifications and reminders are sent to users and restaurant staff.

Comprehensive Reviews:

Detailed restaurant profiles with photos, menus, hours of operation, and reviews.

Users can leave ratings and reviews to share their dining experiences with others.

Advanced Filters:

Users can filter restaurants based on crowdness (busy or quiet), popularity (trending or hidden gems), atmosphere (casual or upscale), and weather conditions (for outdoor seating).

User Profiles and Recommendations:

Personalized user profiles allow for saved preferences, favorite restaurants, and past order history.

The platform uses machine learning algorithms to provide tailored restaurant recommendations based on user behavior and preferences.

Risks and Assumptions:

Market Adoption: Risk of low initial user adoption due to competition from existing platforms.

Reliability of Data: Assumption that restaurant information (availability, menu, reviews) is accurate and up-to-date.

Technical Challenges: Risks associated with integrating multiple APIs for food delivery, map, and weather data.

# **3. Content**

## 3.1 Product vision, roadmap, prototype design

Product vision:

FOR a resident or a tourist WHO is seeking of nice dining place THE Dining Explorer is a Web-based service and an Application THAT provides a chance for users to explore, choose, and enjoy dining experiences tailored to their preferences while providing restaurants with a platform to showcase their offerings and connect with customers. UNLIKE other services and applications OUR product provides an opportunity to find, rate, locate, and even use a delivery service on one platform (Bass, L., 2012).

Roadmap (fig.1.).

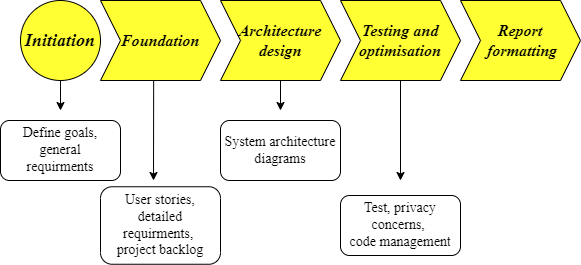


Fig.1. Project roadmap

The design prototype of a mobile version of the application (fig.2.) consists main part of searching box with filter options, after filtering and finding the appropriate variant will be available opportunities to make a reservation or a delivery order.



Fig.2. Prototype design

## 3.2 Agile methodology employed (Cockburn, A., 2001)

Project Kickoff (20th February - 21st February)

• Define project goals, scope, and requirements.

• Create initial user stories and backlog.

• Assign roles and responsibilities within the team.

Sprint Planning for Sprint 1 (22nd February - 23rd February)

• Conduct sprint planning meetings to select user stories for the first sprint (1-week duration).

• Break down user stories into tasks and estimate effort.

• Define sprint goals and acceptance criteria.

Sprint 1 (24th February - 1st March)

• Implement user stories selected for the sprint.

• Conduct a weekly Scrum meeting to discuss progress, obstacles, and plan for the next week.

• Iteratively test and validate features.

Sprint Review and Retrospective (2nd March - 3rd March)

• Review sprint with potential users and gather feedback.

• Conduct a sprint retrospective to discuss what went well, areas for improvement, and action items for the next sprint.

Sprint Planning for Sprint 2 (4th March - 5th March)

• Select user stories for the second sprint based on feedback and priorities.

• Break down user stories into tasks and build system architecture’s diagrams (Martin, R. C., 2008).

Sprint 2 (6th March - 20th March)

• Build use-case, class, sequence and state diagrams.

• Conduct weekly Scrum meetings to track progress and address any issues.

Sprint Review and Retrospective (21th March - 23th March)

• Review sprint 2 deliverables, gather feedback, and conduct a retrospective.

• Identify areas for improvement.

Sprint Planning for Sprint 3 (24th March - 25th March)

• Select user stories for the third sprint

• Break down user stories into tasks and implement them into testing and privacy concern insights.

Sprint 3 (24th March – 7th April)

• Testing, privacy concerns and code management

• Conduct sprint Scrum meetings to track progress and adjust plans as needed.

Sharing results and reworking details of the project (8th April - 15th April)

• Review results after sprint 3, gather final feedback and conduct a retrospective.

• Prepare for final improvements and formalisation.

Making the Assignment (15th April – 21st April)

• Conduct final testing and quality assurance.

• Prepare documentation.

Making the Presentation (22nd April – 26th April)

• Create a great presentation.

• Present our work.

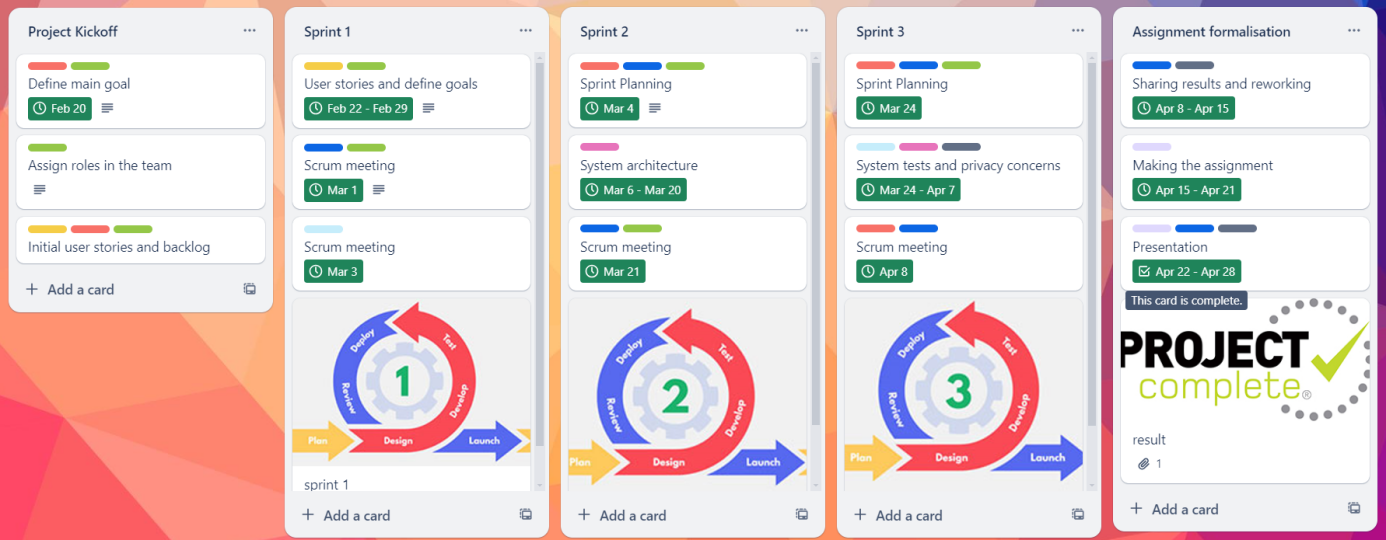


Fig.3. Agile methodology

## 3.3 Features, scenarios, (user) stories, and product backlog

User Story 1: Discovering New Restaurants, Make a reservation

Persona: Food Enthusiast (Emily). She is a mid-age bank manager with an economy degree. She is a foodie and manages her routine life with finding new eating places. As for her constant life exploring new food places becoming to be an adventure and favourite hobby.

During her work she faces some technological tools and dealing with new software is common for her.

Scenario (Cohn, M., 2004):

"As Emily, I want to discover new restaurants based on my preferences, so that I can explore different dining options by using advanced filters to find restaurants based on specific criteria, so that I can choose the perfect dining spot and I want to make table reservations at restaurants to make plans and secure dining experiences in advance."

User requirements:

• Emily opens the Dining Explorer app and selects her preferred cuisine types (Italian and Mexican) and atmosphere (casual).

• She browses through restaurant suggestions with detailed profiles, including photos, menus, user reviews, and ratings.

• She applies advanced filters and discovers a hidden gem with a quiet ambiance and outdoor seating, perfect for a relaxed dining experience.

• Emily saves her favourite filters for future searches, ensuring a personalized and tailored dining exploration.

• She selects the desired date, time, and number of guests, and the app confirms her reservation with a notification.

Features:

• Preference-Based Recommendations: The app suggests restaurants based on user preferences such as cuisine type, price range, and atmosphere.

• Detailed Restaurant Profiles: Detailed profiles for each restaurant including photos, menus, user reviews, and ratings.

• Search and Filter Options: Users can search for specific cuisines or filter restaurants based on various criteria like ambiance and price range.

• Advanced Filter Options: Advanced filters for crowdness, atmosphere, outdoor seating, and other preferences.

• Real-Time Table Availability: Users can check real-time availability for tables at restaurants.

• Reservation Management: Easy reservation management with options to select the date, time, number of guests, and special requests.

• Prepayment: The user confirms his reservation by making the prepayment according to the restaurant's rules.

UML diagrams represent the main requirements and the operation of the system (McConnell, S., 2016).

User Emily wants to find a restaurant applying different filters and make a reservation in the chosen one (fig.4). The search is provided by the main features in the system and the reservation is split into 2 stage processes, firstly should be provided payment by the payment system, and then reservation management confirms the booking. The use case Template is in Appendix B.

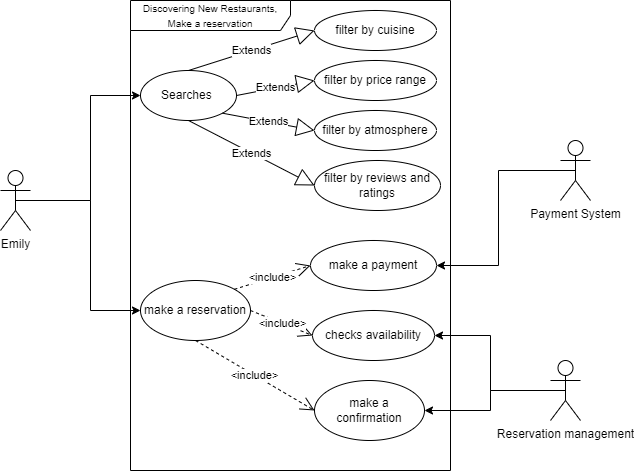


Fig.4. Discovering new restaurant use case

The state chart of the scenario (fig.5). After the software opens and enters the startup state, the user searches for the restaurant, enters the restaurant search state, selects a restaurant, and the software enters the restaurant confirmation state. Save Preference Advanced information saving status. After the user selects the restaurant time, and the number of diners, and pays the advance payment, the restaurant enters the restaurant confirmation state. In this state, the restaurant will give the reservation information to the user, and then the end.

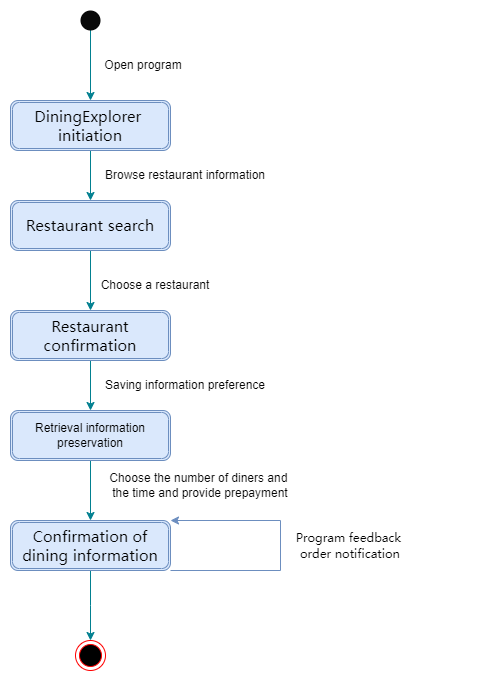


Fig.5. State chart of discovering a new restaurant

User Story 2: Food Delivery Ordering

Persona: Busy Professional (John). He is a 50-year-old senior manager in a big company. He has a few degrees from different prestigious universities and his work mainly connected with communicating with people. It’s important for him not to waste time for anything that is not connected with work.

John is absolutely lost for digital world, that’s why he prefers very user-friendly applications, even though they might be not the best one.

Scenario:

"As John, I want to order food for delivery directly from the app, so that I can enjoy meals at home or work."

User requirements:

• John uses the Dining Explorer app during his lunch break to search for nearby restaurants offering delivery services.

• He selects a restaurant with his favorite dish (sushi) and places an order seamlessly through the integrated food delivery service.

• John receives his sushi order promptly at his office, making his lunch break convenient and enjoyable.

Features:

• Integrated Food Delivery: Seamless integration with food delivery services like Grab and FoodPanda for direct ordering from the app.

• Menu Browsing: Users can browse restaurant menus, select items, and place orders for delivery.

• Order Tracking: Real-time order tracking with updates on order status, estimated delivery time, and delivery partner details.

John firstly searches for a restaurant that suits his cuisine preferences. Then, an actor makes an order with these stages: browsing the menu of the restaurant, selecting items, providing the payment with the payment system, and after that placing the order in the delivery system with a confirmation from it.

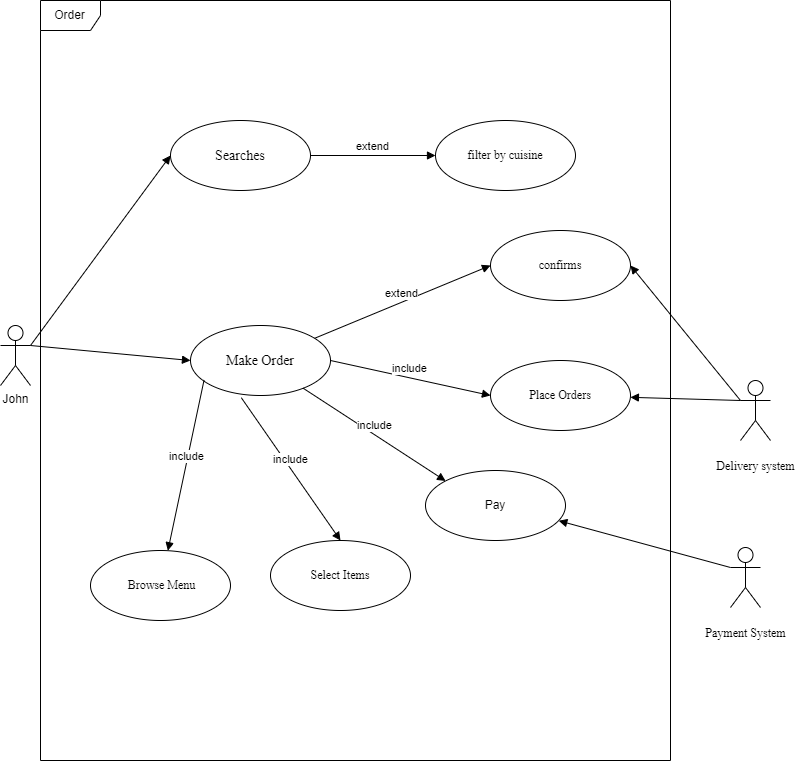


Fig.6. Ordering food use case

The scenario is represented in the state chart as well (fig.7). Users open the application, enter the open state, browse the restaurant information, and enter the restaurant search state. The user selects the restaurant and food and enters the restaurant confirmation state. The user selects the delivery service and enters the distribution status, where the software sends and updates the order information to the user.

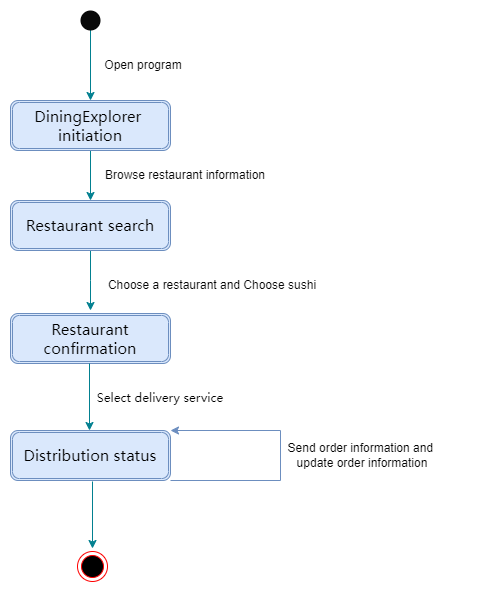


Fig.7. State chart of food delivery orders

User Story 3: Tourist Mode

Persona: Tourist Explorer (Alex). Alex is a young man, who is working remotely for his IT company, which allows him to travel and experience a new life. He likes to use unusual services with unrepeatable features.

As an IT specialist, Ales is a high-advanced user in every service.

"As Alex, a tourist user, I want to explore authentic local dining experiences, so that I can discover the city's culinary scene."

Scenario:

• Alex activates the "Tourist Mode" feature on Dining Explorer upon arriving in a new city.

• He selects interests such as local cuisine, cultural experiences, and popular landmarks.

• The app suggests hidden gems and must-try dishes, guiding Alex to unique dining spots that showcase the city's culinary diversity.

Features:

• Tourist Mode: Specialized mode for tourists with curated recommendations for authentic local dining experiences.

• System of Preferences: Recommendations based on tourist interests such as local cuisine, cultural experiences, and popular landmarks.

• Search and Filter Options: Users can search for specific cuisines or filter restaurants based on various criteria like ambiance and price range.

The user Alex uses “tourist mode” with the main option of the interest (fig.8). Additionally, the searching is provided with specific filters like the atmosphere filter, price filter, and reviews filter.

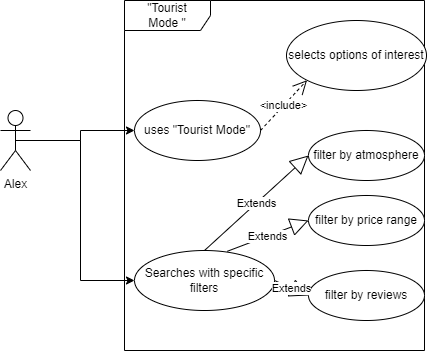


Fig.8. Tourist mode use case

The state chart (fig.9) provides an alternative way to explore this scenario. After the user opens the application and enters the dining explorer initiation, selects the tourist mode and enters the tourist pattern. The user selects the preferences and enters the recommended status. The user uses the search and filter function to submit the restaurant selection status. After that, he selects a restaurant and enters the restaurant confirmation status.

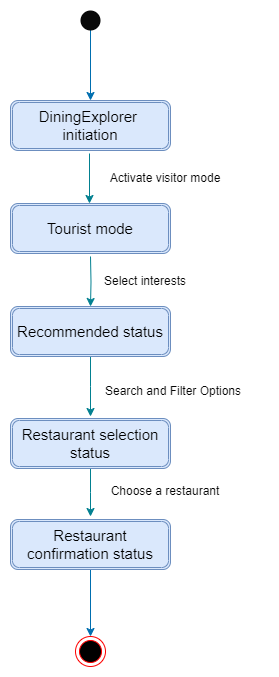


Fig.9. State chart of tourist mode

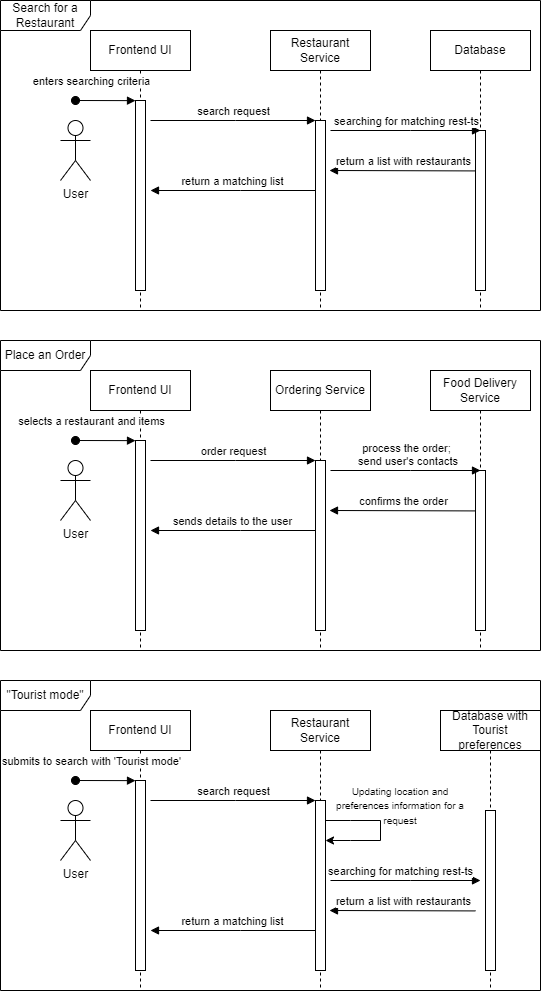


Fig.10. Sequence diagrams of the working system

The sequence diagrams (fig.10) describe 3 previous scenarios. It shows exact interactions between different parts of the software. As can be seen, the most important part is the Frontend User Interface (UI) where the user interacts with the system and submits all of his requests. Then the Frontend UI makes its request to correspond feature to solve the user’s task. The next step for the system is to use the correct service, it could be either the Database, the Delivery System, or any other external API, that is implemented inside the software. At the ending stage of the process, the system responds to the user through the participated feature.

## 3.4 Software architecture

Software architecture is presented by class diagram and microservices inside the system (Brooks, F. P., 1995).

When users search for restaurants, they have two options. Firstly, they can input preferences such as cuisine type, price, address, takeaway availability, current restaurant traffic, popularity, ambiance, outdoor seating, and user ratings to receive a recommended list of restaurants. Secondly, they can opt for "Tourist Mode," where the app automatically suggests local specialty restaurants.

Once users select a preferred restaurant, they can make table reservations and payments within the app. If the restaurant offers takeaway services, users can also place orders, make payments, and choose delivery platforms.

After dining, users can leave reviews and rate their dining experience.

Finally, the app's user profile system analyses user preferences and stores them for future use, allowing for precise restaurant recommendations upon the user's next visit (Beck, K., 2000).

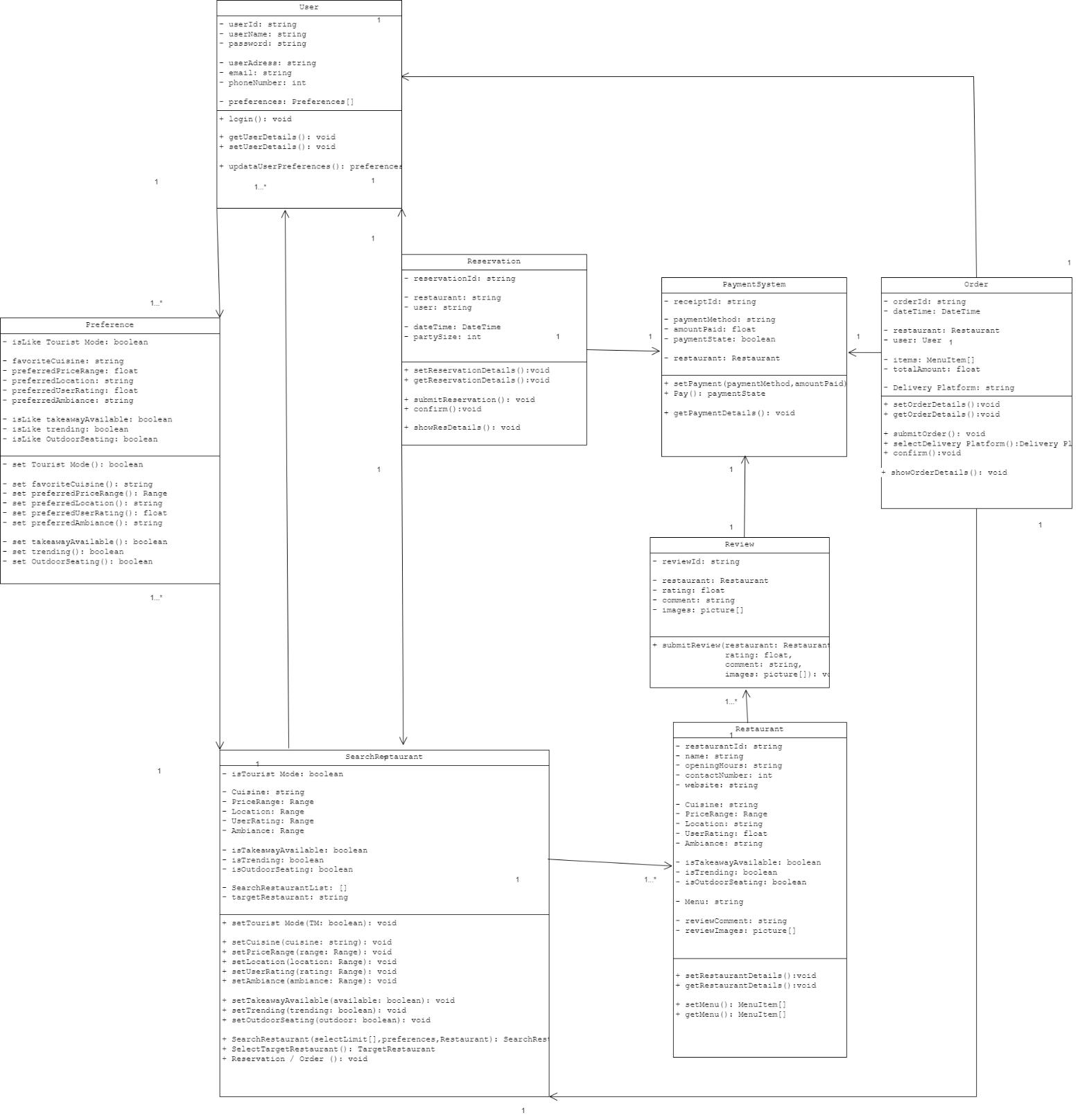


Fig.11. Class diagram

The restaurant recommendation software is designed with the following eight classes to implement its functionality: User, Restaurant, SearchRestaurant, Order, Reservation, Payment, Review, and Preference (Pressman, R. S. (2014).

User Class: This class represents the software's users and manages user information. Users can register accounts, log in to the system, and use various features of the software. Users can search for restaurants, view restaurant information, make table reservations, order takeaways, make payments, and write reviews.

Restaurant Class: This class represents the restaurant entity in the software. Each restaurant object contains information about a specific restaurant, such as name, address, menu, etc. Users can view detailed information about restaurants and make takeaway or table reservation requests as needed. The SearchRestaurant class also depends on the restaurant class for restaurant information during restaurant searches.

SearchRestaurant Class: This class provides the functionality to search for restaurants. Users can search for restaurants based on keywords, location, cuisine, etc. The SearchRestaurant class can interact with the Restaurant class to return a list of restaurants that match the search criteria. It can also interact with the Preference class for preference collection during searches. Additionally, it holds entities for table reservations and takeaway orders for invocation.

Order Class: This class allows users to place takeaway orders in the software. Users can select restaurants, browse menus, add desired items and quantities, and specify delivery platforms. The Order class is responsible for creating, confirming, and delivering orders to delivery platforms. It can invoke instances of the Payment class to complete takeaway order payments.

Reservation Class: This class enables users to make table reservations at restaurants. Users can select restaurants, specify dining dates and times, choose the number of seats, etc. The Reservation class is responsible for handling reservation requests and providing users with confirmation information and reservation details. It can also invoke instances of the Payment class to complete reservation order payments.

Payment Class: This class handles the payment functionality for users. When users successfully place orders or make table reservations, the Payment class is responsible for handling the payment process, including selecting payment methods and payment processing.

Review Class: This class is used for users to provide reviews and feedback on restaurants. Users can write reviews, give ratings, upload images, and share their dining experiences. The Review class can be associated with the Restaurant class to link reviews to specific restaurants.

Preference Class: This class is used to collect and manage user preference settings. The Preference class stores this preference information and uses it in the search and recommendation processes. Each search interacts with preference collection.

These classes work together to enable the restaurant recommendation software to provide features such as user registration, login, restaurant search, takeaway ordering, table reservation, payment, review, and preference collection, providing users with a convenient restaurant selection and dining experience.

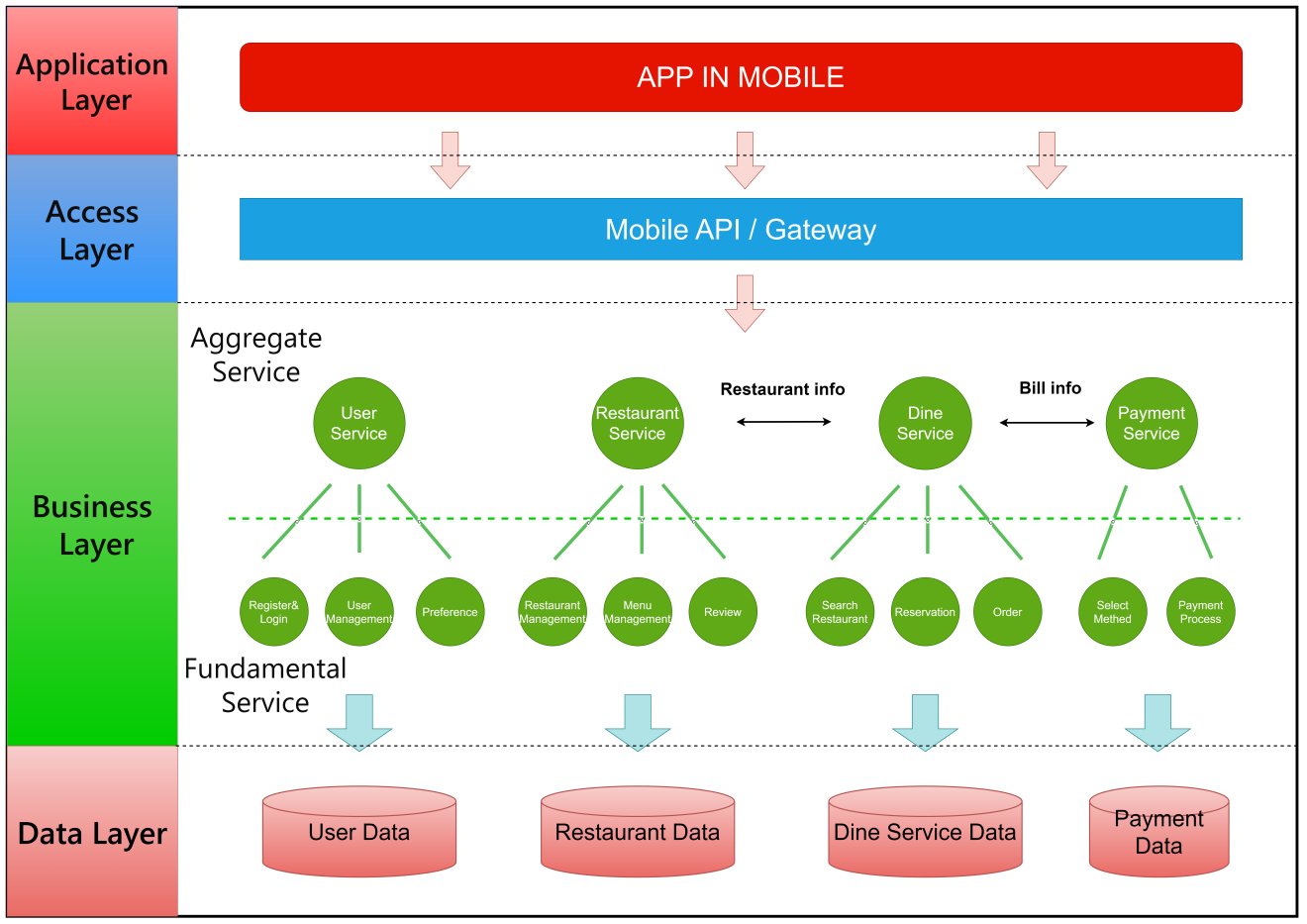


Fig.12. Microservices diagram (Sommerville, I., 2011).

Microservices Architecture is a service-orientated architecture. Microservices Architecture Diagram breaks down a large app into multiple small, independent microservices from the perspective of system services. Each service possesses its own relatively independent business logic and dataset. Such an architecture enables each microservice to exhibit characteristics of single responsibility, loose coupling, and high cohesion, thereby promoting a more modular and scalable system.

The microservices architecture of this software is divided into the following layers:

* Application Layer: Serves as the interface for user-system interaction, sending requests and receiving responses.
* Access Layer: Acts as an intermediary between the mobile app and the domain layer, handling request validation, routing, and forwarding, coordinating calls to domain services.
* Business Layer:
* User Service: Responsible for handling user-related logic, such as user registration, login, and user profile management.
* Restaurant Service: Responsible for handling restaurant-related logic, such as restaurant information retrieval, menu management, reviews, etc.
* Dining Service: Responsible for handling dining-related logic, such as restaurant search, table reservation, takeaway, etc.
* Payment Service: Responsible for handling payment-related logic, such as selecting payment methods, payment order management, etc.
* Data Layer: Independently stores datasets related to each microservice (McConnell, S., 2016).

## 3.5 Security and privacy concerns

For User Data Protection should be (Leffingwell, D., & Widrig, D., 2003):

* Ensured that user data, including personal information, payment details, and account credentials, are stored securely.
* Implemented encryption mechanisms (e.g., SSL/TLS) to protect data transmission over the network.
* Used secure authentication methods (e.g., OAuth, JWT tokens) for user login and session management.

For Payment Security should be:

* Adhered to Payment Card Industry Data Security Standard (PCI DSS) compliance for handling payment information.
* Used tokenization and encryption for storing and processing payment card details.
* Implemented fraud detection and prevention measures to detect and mitigate unauthorized transactions.

Access Control should be:

* Implemented role-based access control (RBAC) to restrict access to sensitive functionalities and data based on user roles.
* Used strong password policies, multi-factor authentication (MFA), and session timeouts to enhance access security.

For Securing APIs and Microservices should be:

* Secured API endpoints with authentication, authorization, and rate-limiting mechanisms.
* Implemented secure communication protocols (e.g., HTTPS, OAuth) for API interactions.
* Monitored and logged API usage for security auditing and anomaly detection.

As a Vulnerability Management the next steps are necessary:

* Conduct regular security assessments, penetration testing, and code reviews to identify and mitigate vulnerabilities.
* Keep software dependencies, libraries, and frameworks up to date with security patches and updates

## 3.6 Test plan, test examples, and results

The testing scenarios are provided in Table. All the testing actions are made by implemented features inside the application.

|  |  |  |
| --- | --- | --- |
| Test input | Process | Test output |
| User login | The account service feature manages the action and checks the user authorization information | User’s account |
| The user searches the restaurant by name | The Search and Filter feature provides an according finding in the database | A list of the restaurants |
| Applying filters for advanced search | The Search and Filter and Advanced Filter features used to form the corresponding list | A list of the restaurants |
| Applying the ‘tourist mode’ | The ‘tourist mode’ feature with corresponding filters | A list of restaurants and preferences |
| Reserve a table | The Reservation Management feature checks the availability of seats and applies the booking | Transferring to the prepayment feature |
| Provide a prepayment | The prepayment/payment feature makes the payment | A receipt and booking evidence |
| Make a delivery | The Integrated Food Delivery feature requests the delivery system | Booking status and order tracking |

## 3.7 DevOps and code management

During the project assessment, we will try to create processes that are easier to audit and that attest to the effectiveness of controls, in support of compliance with regulatory and contractual obligations (Kim, G. et al., 2021). We do this by (Fowler, M., 1999):

* Making security a part of everyone’s job
* Integrating preventative controls into our shared source code repository
* Integrating security with our deployment pipeline
* Integrating security with our telemetry to better enable detection and recovery
* Protecting our deployment pipeline
* Integrating our deployment activities with our change approval processes
* Reducing reliance on separation of duty

Continuous delivery includes creating the foundations of our automated deployment pipeline, ensuring that we have automated tests that constantly validate that we are in a deployable state, having developers integrate their code into the trunk daily, and architecting our environments and code to enable low-risk releases (Fowler, M., 1999). Primary focuses include:

* Creating the foundation of our deployment pipeline
* Enabling fast and reliable automated testing
* Enabling and practicing continuous integration and testing
* Automating, enabling, and architecting for low-risk releases

Our goal is to make it possible to see and solve problems as they occur, and we grow safe systems of work that allow us to confidently make changes and run product experiments, knowing we can quickly detect and remediate failures. We will do all of this by implementing the following:

* Integrating user research and feedback into the work of product teams
* Enabling feedback so Dev and Ops can safely perform deployments
* Enabling feedback to increase the quality of our work through peer reviews and pair programming

# **4. Conclusion**

In conclusion, the Dining Explorer project represents a transformative endeavor in the realm of dining experiences. Through innovative features such as restaurant discovery, food delivery integration, table reservations, and user reviews, the platform aims to revolutionize how users explore, enjoy, and engage with culinary offerings.

Key findings from the project include the successful implementation of user-centric design, seamless integration with third-party services, and the establishment of a vibrant community of food enthusiasts. These achievements highlight the project's ability to deliver value, enhance user experiences, and foster meaningful connections within the dining ecosystem.

Moving forward, the project's outcomes suggest promising opportunities for further innovation, collaboration with industry partners, and continuous improvement in meeting user needs and expectations. By leveraging these insights and lessons learned, the Dining Explorer project sets a solid foundation for ongoing success and future growth in the dynamic landscape of dining exploration and enjoyment.

# **5. Work breakdown by each member written for the rest of the members**

|  |  |
| --- | --- |
| Name of the member | Completed tasks |
| Timur Mamadaliyev | Product vision, roadmap, prototype design  Agile methodology  Features, scenarios, (user) stories and product backlog  Software architecture  Security and privacy concerns  Test plan, test examples, and results  DevOps and code management  Conclusion  References |
| Thai | Software architecture  References |
| Zhang ShanShan | Features, scenarios, (user) stories and product backlog |
| Yang MinXin | None |

# **6. References**

Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice (3rd ed.). Boston, MA: Addison-Wesley.

Beck, K. (2000). Extreme Programming Explained: Embrace Change (2nd ed.). Boston, MA: Addison-Wesley.

Booch, G. et al. (2005) The unified modeling language user guide. Second edition. Upper Saddle River, NJ: Addison-Wesley.

Brooks, F. P. (1995). The Mythical Man-Month: Essays on Software Engineering (Anniversary ed.). Reading, MA: Addison-Wesley*.*

Cockburn, A. (2001). Agile Software Development: The Cooperative Game (2nd ed.). Boston, MA: Addison-Wesley.

Cohn, M. (2004). User Stories Applied: For Agile Software Development. Boston, MA: Addison-Wesley.

Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (1999). Refactoring: Improving the Design of Existing Code. Boston, MA: Addison-Wesley.

Fowler, M. (1999). Refactoring: Improving the Design of Existing Code. Boston, MA: Addison-Wesley.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley.

Kim, G. et al. (2021) The devops handbook how to create world-class agility, reliability, & Security in Technology Organizations. Portland, OR: IT Revolution Press, LLC.

Leffingwell, D., & Widrig, D. (2003). Managing Software Requirements: A Unified Approach. Boston, MA: Addison-Wesley.

Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River, NJ: Prentice Hall.

McConnell, S. (2016). Code Complete: A Practical Handbook of Software Construction (2nd ed.). Redmond, WA: Microsoft Press.

Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach (8th ed.). New York, NY: McGraw-Hill Education.

Sommerville, I. (2011). Software Engineering (9th ed.). Boston, MA: Addison-Wesley.

# **Appendix A. Scrum meeting description**

The Scrum meetings description is provided in Table

|  |  |  |
| --- | --- | --- |
| Name of the Scrum | Minutes spend on meeting | Description of the meeting |
| Project Kickoff | 60 | Define project goals,  Assign roles and responsibilities |
| Sprint Planning for Sprint 1 | 90 | Break down user stories into tasks.  Define sprint goals and acceptance criteria |
| Sprint 1 (Review and Retrospective) | 90 | Review sprint with potential users and gather feedback  Discuss what went well, areas for improvement, and action items for the next sprint |
| Sprint Planning for Sprint 2 | 60 | Planning to build system architecture’s diagrams |
| Sprint 2 (Review and Retrospective) | 90 | Review sprint 2 deliverables, gather feedback, and conduct a retrospective.  Identify areas for improvement |
| Sprint Planning for Sprint 3 | 60 | Work with testing and privacy concern insights. |
| Sharing results and reworking details of the project | 120 | Review the whole work and manage to provide the necessary rework |

# **Appendix B. Use case Templates**

|  |  |
| --- | --- |
| **USE CASE DESCRIPTION** | |
| **SYSTEM** | **Dining Explorer** |
| **USE CASE NAME: Discovering New Restaurants, make a reservation** | |
| **PRIMARY ACTORS**  **Emily** | **OTHER ACTORS**  **Payment System, Reservation management** |
| **STAKEHOLDERS** | **Emily** |
| **DESCRIPTION**  **User Emily wants to find a restaurant applying different filters and make a reservation in the chosen one. The search is provided by the main features in the system and the reservation is split into 2 stage processes, firstly should be provided payment by the payment system, and then reservation management confirms the booking.** | |
| **RELATIONSHIPS**  **INCLUDES: make a payment, check availability, make a confirmation**  **EXTENDS: filter by cuisine, filter by price range, filter by atmosphere, filter by reviews and rating** | |
| **INPUT: Searches, make a reservation** | |
| **PRE-CONDITIONS: user login/registration** | |
| **STEPS: Searches, extends; make a reservation, includes** | |
| **ACTOR** | **SYSTEM** |
| **Emily** | **Searching and Filter option feature** |
| **Payment system** |  |
| **Reservation management** |  |
| **ALTERNATIVES AND EXCEPTIONAL FLOWS:**  **If the payment is not provided, the reservation is not going to be confirmed** | |
| **POST CONDITIONS** | |

|  |  |
| --- | --- |
| **USE CASE DESCRIPTION** | |
| **SYSTEM** | **Dining Explorer** |
| **USE CASE NAME: Order** | |
| **PRIMARY ACTORS**  **John, Delivery system** | **OTHER ACTORS**  **Payment System** |
| **STAKEHOLDERS** | **John** |
| **DESCRIPTION**  **John firstly searches for a restaurant that suits his cuisine preferences. Then, an actor makes an order with these stages: browsing the menu of the restaurant, selecting items, providing the payment with the payment system, and after that placing the order in the delivery system with a confirmation from it.** | |
| **RELATIONSHIPS**  **INCLUDES: browse menu, select items, pay, place orders**  **EXTENDS: filter by cuisine, confirms** | |
| **INPUT: searches, make an order** | |
| **PRE-CONDITIONS: user’s login/authorisation** | |
| **STEPS: searches, filter by cuisine; make an order, browse, select, pay, place, confirms** | |
| **ACTOR** | **SYSTEM** |
| **John** | **Search and filter feature** |
| **Delivery System** |  |
| **Payment System** |  |
| **ALTERNATIVES AND EXCEPTIONAL FLOWS:**  **If the payment is not provided, the delivery order is not going to be confirmed** | |
| **POST CONDITIONS** | |

|  |  |
| --- | --- |
| **USE CASE DESCRIPTION** | |
| **SYSTEM** | **Dining Explorer** |
| **USE CASE NAME: Tourist mode** | |
| **PRIMARY ACTORS**  **Alex** | **OTHER ACTORS** |
| **STAKEHOLDERS** | **Alex** |
| **DESCRIPTION**  **The user uses “tourist mode” with the main option of the interest. Additionally, the searching is provided with specific filters like the atmosphere filter, price filter, and reviews filter.** | |
| **RELATIONSHIPS**  **INCLUDES: selects options of interest**  **EXTENDS: filter by atmosphere, filter by price range, filter by reviews** | |
| **INPUT: uses Tourist mode, searches with specific filters** | |
| **PRE-CONDITIONS: user’s login/authorization** | |
| **STEPS: uses Tourist Mode, selects options; searches with specific filters, extends** | |
| **ACTOR** | **SYSTEM** |
| **Alex** | **‘Tourist mode’ feature** |
|  | **Search and Filter options features** |
| **ALTERNATIVES AND EXCEPTIONAL FLOWS:**  **If no filters are applied, the tourist mode will provide a list of restaurants according only to the interest option.** | |
| **POST CONDITIONS** | |

# **Appendix C. Turnitin report**

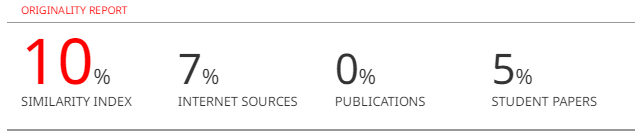


Fig.13. Turnitin report