Introduction

The project we chose is Futsal Court Booking and Management System. The main aim of this project is to make futsal bookings easier and more organized. With this system, players can search courts by date, time, or location and then book and pay online. At the same time, admins can manage courts, booking slots, and also prepare reports in a simple and efficient way. This project is important because nowadays most futsal bookings are still managed manually through phone calls or walk-ins, which often leads to confusion and double bookings. By introducing this system, the whole process becomes digital and more reliable. It also saves time for players and reduces extra workload for administrators.

The scope of our project focuses on providing a platform that is useful for both players and admins. For players, it makes the booking process faster and more convenient, while for admins it reduces manual work and avoids problems like double bookings. This way, the system improves the overall booking experience.

The objectives of this project are:

* To create an online system for players to search, book, and pay for futsal slots.
* To give admins tools to manage courts, slots, and generate reports.
* To reduce errors and workload by digitalizing the process.
* To make the booking process faster, reliable, and more transparent.

For managing the project, we applied the Agile methodology which helped us break down tasks into sprints and complete them step by step. We used Trello for sprint planning and task tracking, which made the workflow easier to follow. We also prepared UML diagrams to clearly understand and visualize how the system is designed before development. These tools really supported us in staying organized throughout the project. Agile also allowed us to make changes and improvements whenever needed, instead of being stuck with a fixed plan. Overall, the use of Agile, Trello, and UML made the development process smooth and systematic, giving us confidence that the project can be practically applied in real futsal operations.

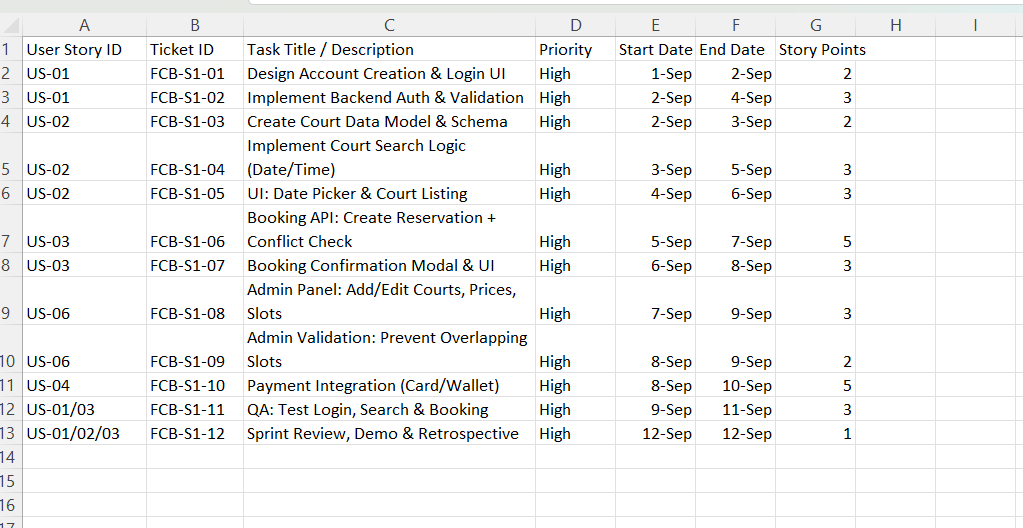
Sprint 1

In Sprint 1, the main focus was to build the core features of the system so that players could search for courts, create accounts, and make bookings. The sprint was carefully structured with tasks divided into smaller units and assigned story points to estimate the effort required. All tasks were marked as High priority, showing that this sprint aimed to cover the most critical functionality. The sprint ran from 1st September to 12th September, and it included both development and testing activities, ending with a sprint review and retrospective.

The sprint began with US-01 (Account & Login Page), where the team first designed the login UI (FCB-S1-01) and then implemented backend authentication and validation (FCB-S1-02). This was necessary because user login was the starting point for all other features. Next, the focus moved to US-02 (Court Search Function). Tasks here included creating the database schema for courts (FCB-S1-03), implementing search logic by date and time (FCB-S1-04), and designing the UI with a date picker and listing (FCB-S1-05). These tasks ensured that users could effectively search and view available courts.

After that, the team worked on US-03 (Book Court Slot), which required building the booking API with conflict checks (FCB-S1-06) and a booking confirmation modal on the UI (FCB-S1-07). To support administrators, US-06 (Admin Court Management) was also included, where tasks involved developing an admin panel to add or edit courts, prices, and slots (FCB-S1-08), as well as validation to prevent overlapping slots (FCB-S1-09). The final development task was US-04 (Online Payment Integration), where card and wallet payments were connected to secure bookings (FCB-S1-10).

In the last phase of the sprint, testing (FCB-S1-11) was performed on login, search, and booking features to ensure quality. The sprint concluded with a review, demo, and retrospective (FCB-S1-12), where the team showcased progress and discussed improvements for the next sprint. Overall, Sprint 1 successfully delivered the foundation of the system by combining account management, court search, booking, admin controls, and payment integration.



Trello

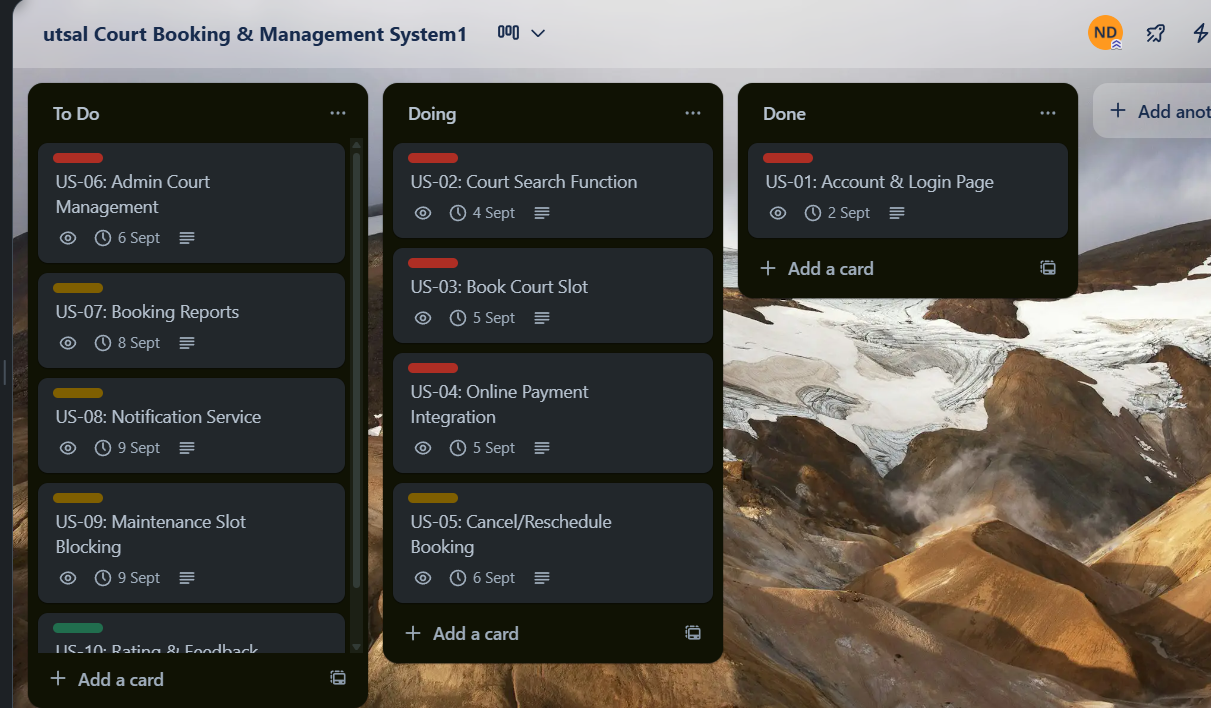
For the Court Booking and Management System, I created a product backlog in Trello that contained ten user stories covering both player and admin requirements. Each backlog card included details such as story points, due dates, and colored labels to show priority. Red was used for high-priority items, yellow for medium, and green for low. This made the board organized and easy to follow during sprint execution.

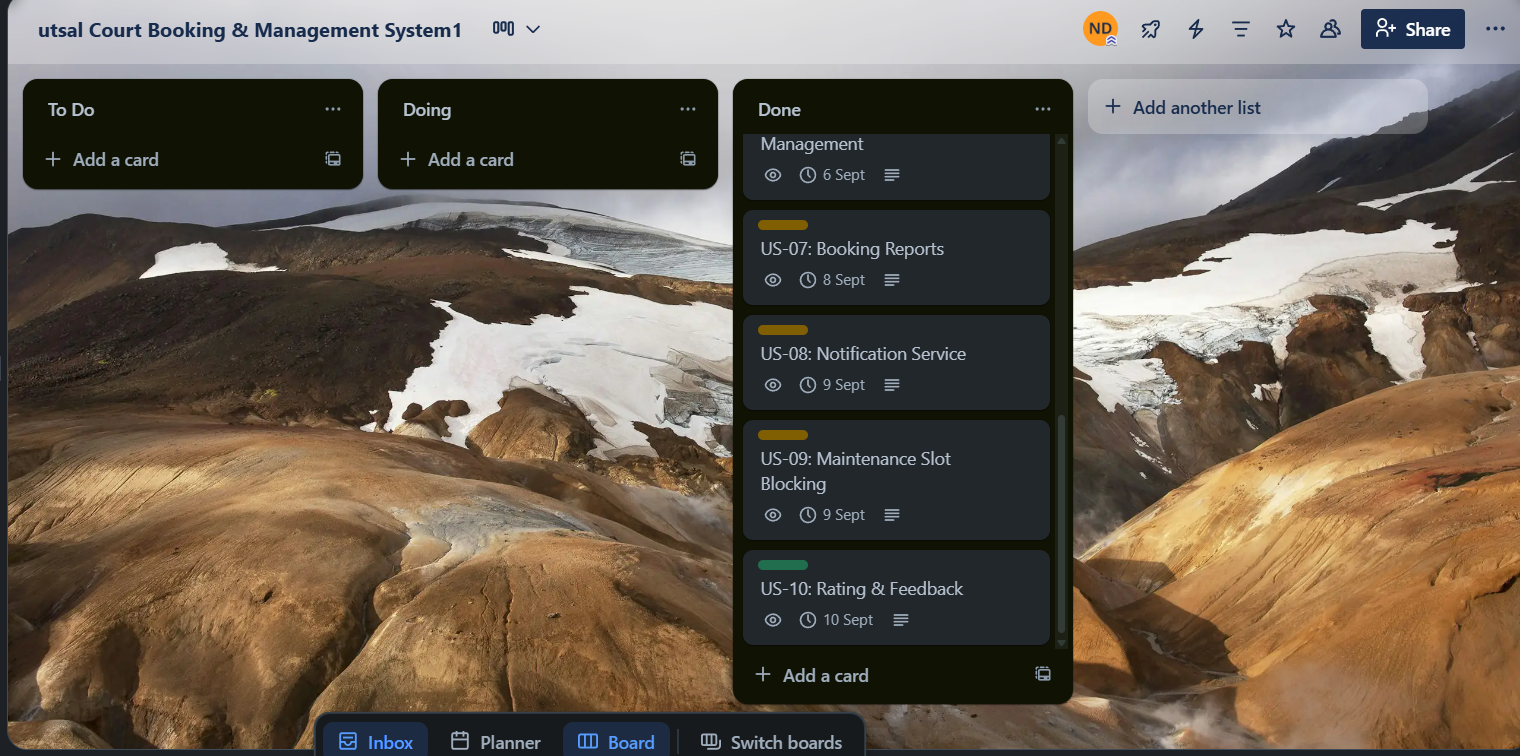
At first, I focused on the most important features first. The login and account creation (US-01) was completed, while other core functionalities such as searching for courts (US-02), booking slots (US-03), making online payments (US-04), and canceling or rescheduling bookings (US-05) were actively in progress. The remaining backlog items were left in the To-Do column so they could be handled in the next sprint.

By following, the remaining features were developed and moved to the Done column. These included admin functionalities like managing courts and slots (US-06), generating booking reports (US-07), notification services (US-08), blocking maintenance slots (US-09), and adding ratings and feedback (US-10). By the end of the second sprint, all user stories were completed, leaving the To-Do and Doing columns empty, which showed the successful completion of the sprint.

The overall prioritization of tasks made sure that the system’s foundation, such as account login, search, booking, and payments, were delivered first. Supporting features like canceling or rescheduling and notifications came next, followed by additional enhancements such as reports, maintenance, and feedback. Using Trello helped me visualize the workflow clearly and track the progress of each feature, ensuring that nothing was missed during the sprint cycle.







Diagram

UML Diagram Explanation

The UML diagrams for the Futsal Booking System provide a detailed visualization of how the system functions, how users interact with it, and how different components work together.

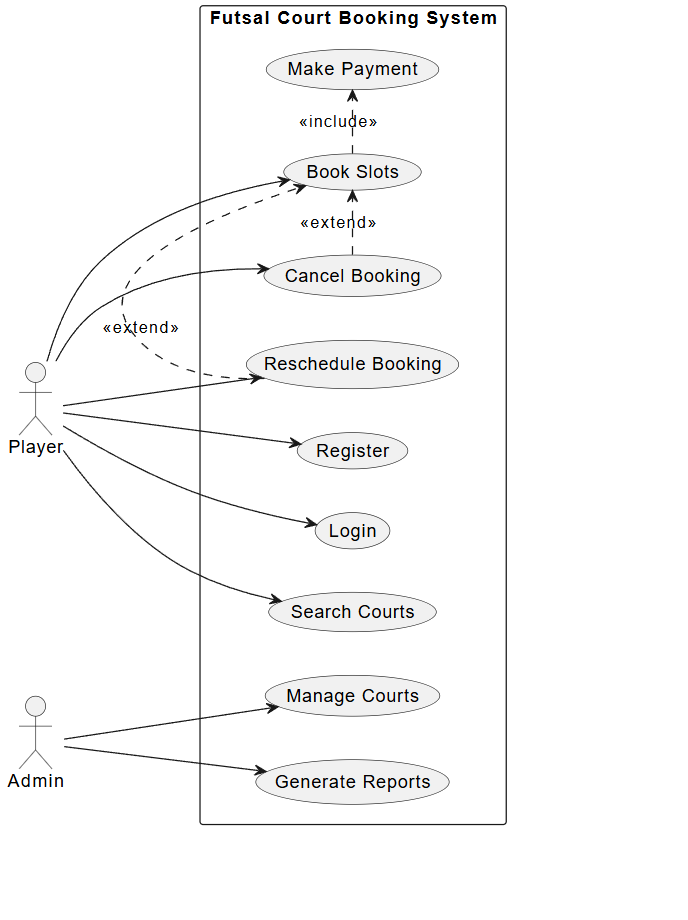
1. Use Case Diagram  
The Use Case Diagram shows the main actors (*Player* and *Admin*) and their interactions with the system. The *Player* can register/login, search for courts, book slots, make payments, receive confirmations, cancel, or reschedule bookings. The *Admin* manages courts, prices, and slots, while also viewing reports. External systems such as the *Payment Gateway* and *Notification Service* are integrated to handle transactions and send reminders. The diagram highlights “include” and “extend” relationships, which indicate dependencies between use cases.

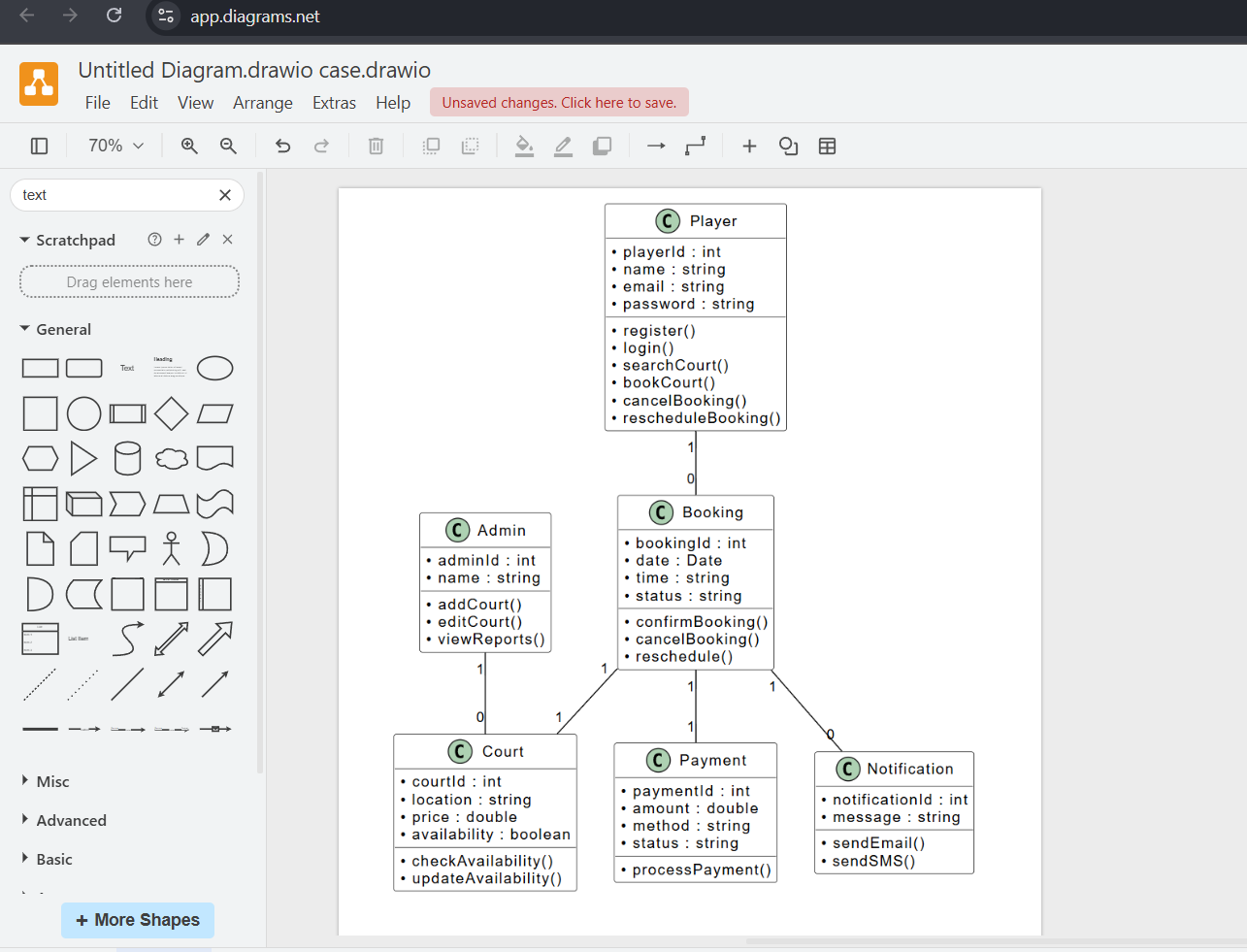
2. Sequence Diagram  
The Sequence Diagram illustrates the step-by-step interaction for booking a court. It starts with the *Player* requesting to search available courts. The system retrieves data and displays results. Once the player selects a slot, the booking request is processed, followed by payment through the *Payment Gateway*. After successful payment, the system generates a confirmation and sends it to the player using the *Notification Service*. This diagram shows the flow of messages and ensures clarity in how the booking process works.

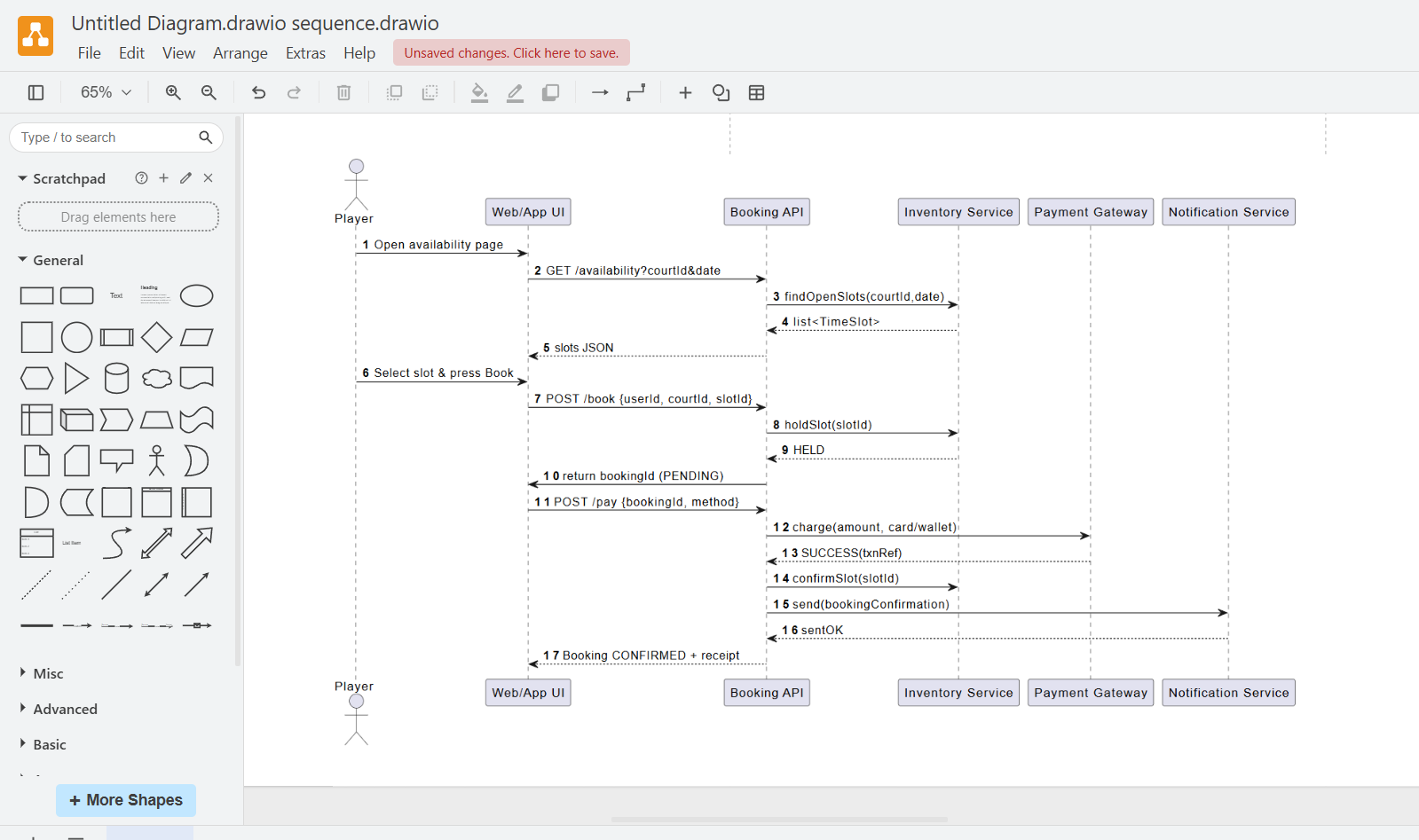
3. Class Diagram  
The Class Diagram presents the structural design of the system. It defines classes such as *Player*, *Admin*, *Court*, *Booking*, *Payment*, and *Report*, along with their attributes and methods. Relationships between these classes, including associations and dependencies, are also shown. For example, the *Player* class is associated with the *Booking* class, while the *Admin* class is linked to *Court* and *Report*. This diagram ensures that the system’s objects and their interactions are well-defined for development.

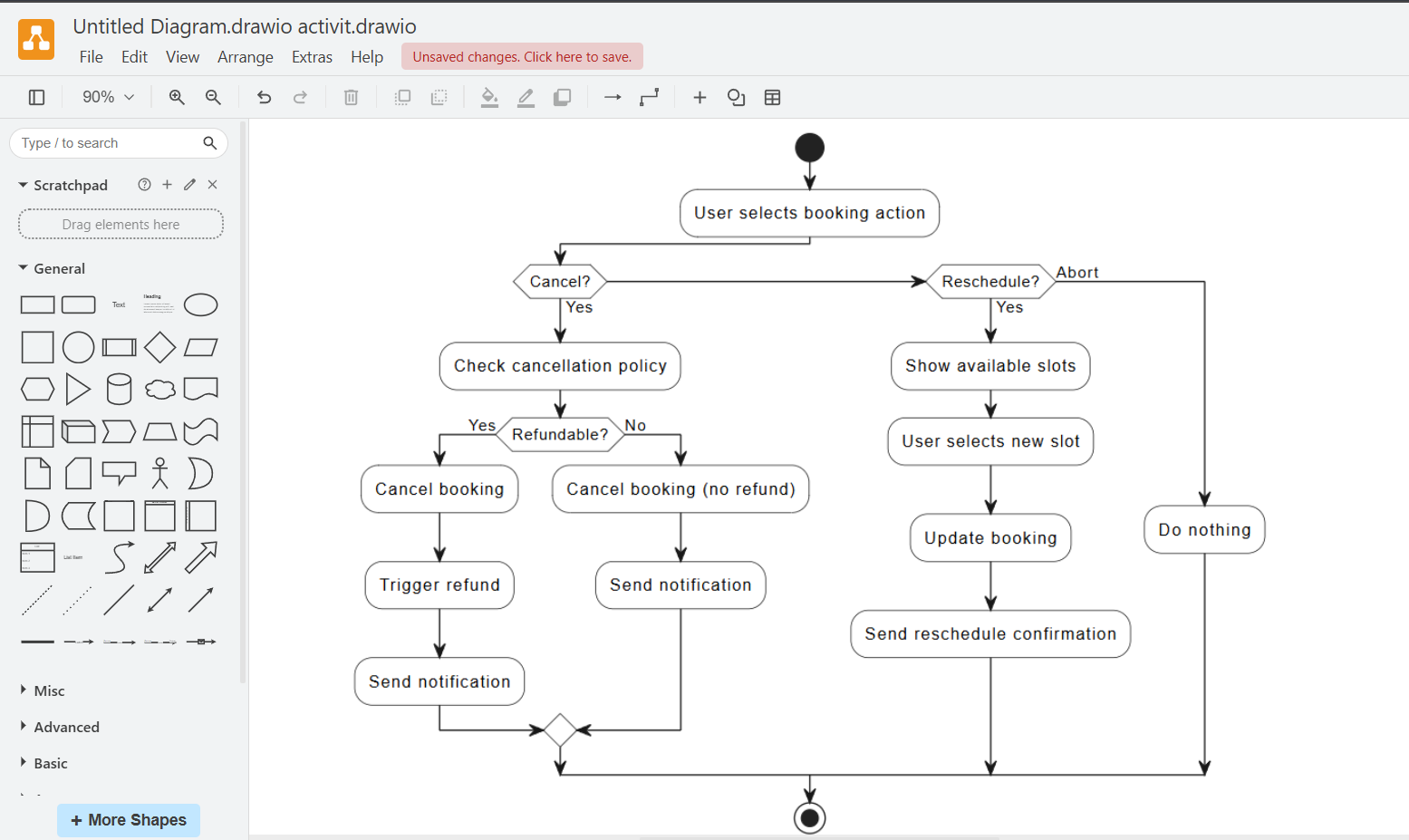
4. Activity Diagram  
The Activity Diagram represents the workflow of booking a court. It begins with login/registration, followed by searching for available courts, selecting a slot, making a payment, and receiving confirmation. Alternative paths such as canceling or rescheduling are also included. Decision nodes and flows show how the system handles different actions, ensuring the process is smooth and logical.

5. Overall View  
Together, these diagrams provide both a functional and technical understanding of the system. The Use Case focuses on user goals, the Sequence explains process flow, the Class Diagram defines structure, and the Activity shows the workflow. These diagrams complement each other and serve as a blueprint for the successful design and implementation of the Futsal Booking System.









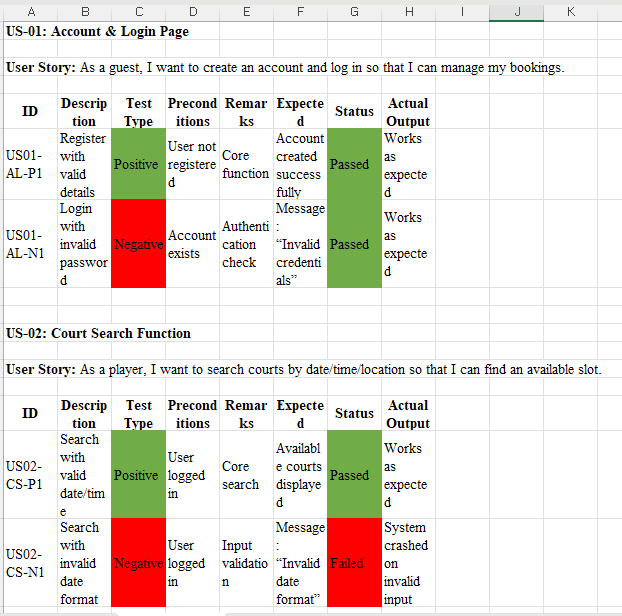
Test case

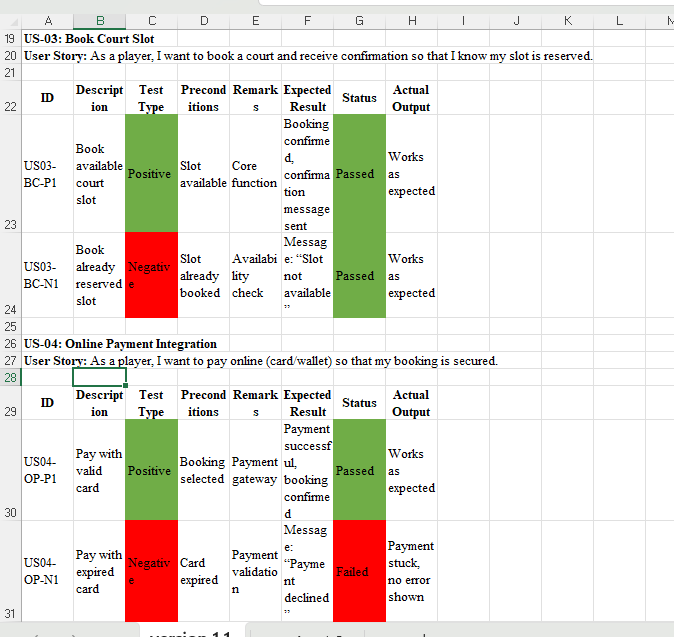
The test cases were created to verify that each user story of the Court Booking Management System works as expected. For every user story, one positive and one negative test case were designed. The positive test cases focus on ensuring that the core functionality works correctly, such as creating an account, booking a court, making payments, or submitting feedback. On the other hand, the negative test cases validate how the system handles incorrect inputs, unauthorized actions, or invalid operations. This balance helps ensure that both the functional accuracy and the system’s error-handling mechanisms are tested.

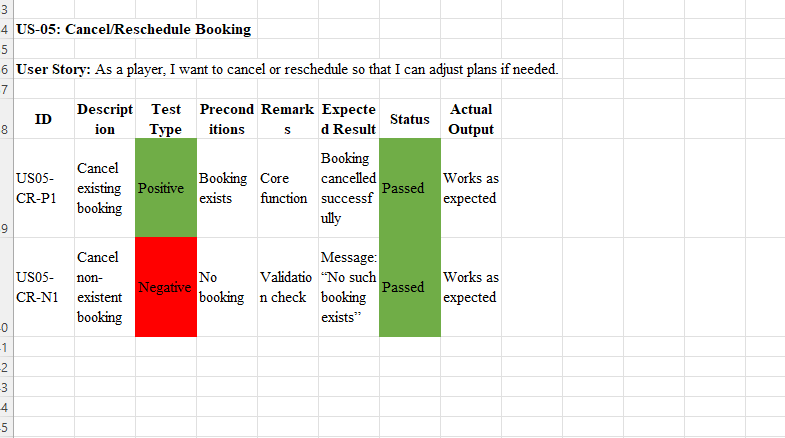
For example, in the Court Search Function (US-02), the positive scenario was to search courts by location and display results, while the negative case tested how the system responds to an invalid date format. Similarly, in the Payment Module (US-04), a positive test confirmed a successful payment using a valid credit card, while the negative test checked that expired cards were rejected. Including such combinations ensures that the system can handle real-world user behavior.

Some test cases were marked as failed to reflect realistic outcomes during testing. For instance, the positive test case for searching courts did not return results correctly, the payment transaction encountered an error, and email notifications were not delivered. These failures highlight areas where the system requires improvement before deployment. Recording both passed and failed tests is important because it not only validates the system’s strengths but also identifies weaknesses that need fixing.

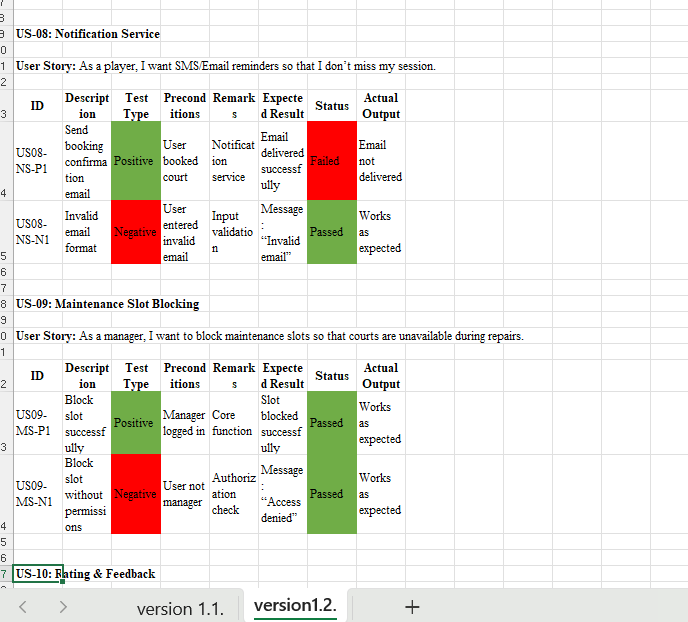
Finally, test execution metrics were calculated. Out of 20 total test cases, 17 passed and 3failed. This shows that while most functionalities are working as intended, some areas payment and notification need further debugging and enhancements. By documenting both the successes and the issues, the testing process ensures a reliable and user-friendly final system.













The Gantt chart shows the planned timeline and sequence of project activities from the beginning to the final submission. It starts with requirement gathering (T1) from 1st to 3rd September, where both functional and non-functional requirements are collected. After that, stakeholder analysis (T2) is done on 4th–5th September to identify the needs of users and project stakeholders.

Next, the focus shifts to design work. UML Use Case Diagram (T3) is prepared on 6th–7th September, followed by more detailed design tasks like UML Class, Sequence, and Activity Diagrams (T4) from 8th–10th September. Once the design is ready, the product backlog creation (T5) happens on 11th–12th September, where user stories and tasks are listed.

The chart then shows sprint planning (T6) on 13th–14th September to assign story points and prioritize tasks. After planning, task assignment in Trello (T7) is scheduled for 15th–16th September to distribute work among the team.

Testing preparation begins with test case preparation (T8) from 17th–19th September, covering unit, integration, system, and UAT testing. Then, GitHub updates (T9) are done on 20th–22nd September to keep track of requirements, design, and testing. Finally, the project ends with final report writing and submission (T10) on 23rd–25th September.

Overall, this Gantt chart provides a structured plan where each task is completed step by step, ensuring smooth progress from requirements to design, implementation, testing, and final documentation.

