MEETING SCHEDULES

December 4th:

Define product backlog

December 5th:

Sprint planning

December 6th:

Daily scrum 1

December 7th:

Daily scrum 2

December 8th:

Daily scrum 3

December 9th:

Sprint Review

Sprint Retrospective

PRODUCT BACKLOG

Epic 1: Basic Scheduling System Functionality

User Story 1:
As an admin, I want to input the details of teams, venues, and schedules so that I can create a league schedule.

Acceptance Criteria:

- Admin can input the number of teams, venue details, and game days.
- System validates input for missing or incorrect data (e.g., invalid team name, incorrect game days).
- Data is stored in a structured format (database or file) for schedule generation.
- Admin receives appropriate error or success messages if input is invalid or valid.

User Story 2:
As an admin, I want to generate a schedule for a league of teams so that games can be evenly distributed.

Acceptance Criteria:

- System generates a schedule that ensures all teams have equal playing time.
- Teams may play more than once on the same day (double-headers).
- The schedule adapts to constraints, such as venue availability and team count (e.g., no team exceeds the maximum number of games in a week).
- The system provides a way to view and validate the generated schedule.

Epic 2: Advanced Scheduling Scenarios

User Story 3:
As an admin, I want to distribute game times and venues evenly across teams for competitive leagues.

Acceptance Criteria:

- Teams alternate between 6 PM and 7 PM slots.
- Teams rotate between two specified venues each week.
- No team plays consecutively in the same time slot or at the same venue unless unavoidable.
- The system ensures that the game schedule adheres to alternating venue and time slots while maintaining fairness across teams.

User Story 4:
As an admin, I want to manage incomplete leagues where fewer teams have signed up so that the league can still run.

Acceptance Criteria:

- The system generates a balanced schedule for fewer teams, ensuring each team plays a reasonable number of matches.
- Unassigned time slots (if fewer teams) are managed equitably (e.g., extra matches or empty slots).
- The system can accept late registrations and fill any scheduling gaps.
- Any late registrants are integrated into the schedule without disrupting the fairness of the previous matchups.

Epic 3: Testing and Reporting

User Story 5:
As a developer, I want unit tests for each module of the scheduling system so that code reliability is ensured.

Acceptance Criteria:

- Unit tests are written for input validation (e.g., missing data, incorrect formats).
- Unit tests for schedule generation logic (e.g., equal distribution of games, double-headers).
- Unit tests for constraints validation (e.g., venue availability).
- Test coverage is at least 80% of the codebase.
- The tests are automated and run consistently in the development environment.

User Story 6:
As a developer, I want to produce a test report showing results of all executed tests so that the testing process is documented.

Acceptance Criteria:

- The test report includes a table listing all tests executed, their execution status (pass/fail), and any critical findings.
- The test report includes a summary of the overall test coverage and any significant bugs or issues identified.
- The report should be easily understandable and provide clear details for the next steps if issues are found.
- The report is shared with the product owner for review.

SPRINT PLANNING

Sprint Goal: Deliver a scheduling algorithm that takes input data from files and gives a schedule for the teams in another output file.

1. **Define Algorithm Requirements**

User Story:
"As a customer, I want to define the requirements for the scheduling algorithm, such as how teams are matched and the constraints (e.g., match day, venue capacity), so that the algorithm can be designed to meet my league's needs."

Acceptance Criteria:

- The requirements for scheduling are clearly documented, including the number of teams, match days, and venue constraints.
- Constraints around team matching, double-headers, and match-day rotation are clearly defined.
- The document must outline the expected output for the scheduling algorithm.
- The product owner has reviewed and approved the requirements.

Definition of Done (DoD):

- A comprehensive document outlining the scheduling algorithm's requirements is created.
- The document includes:
    - League types (e.g., recreational, competitive).
    - Constraints (e.g., match day, number of games per team).
    - Rules for handling varying team counts and late registrations.
- The document is reviewed and approved by the product owner.

---

2. Test the Scheduling Algorithm with Sample Scenarios

User Story:
"As a customer, I want to see the scheduling algorithm tested with sample league scenarios (e.g., 12 teams playing double-headers, 10 teams in a competitive league), so that I can ensure it works correctly for different types of leagues."

Acceptance Criteria:

- The algorithm has been tested with at least three predefined sample scenarios.
- For each scenario, the output matches the expected schedule.
- The scenarios should include different configurations (e.g., number of teams, venue constraints).
- No errors or major discrepancies are present in the results.
- The product owner confirms that the scenarios have been tested correctly.

Definition of Done (DoD):

- The algorithm has been executed using at least 3 sample scenarios (e.g., 12 teams with double-headers, 10 teams with alternating venues).
- The generated schedule for each scenario is reviewed for correctness.
- Any issues found during the testing phase are logged for investigation.
- The product owner has reviewed the output schedules and confirmed they meet the requirements.

---

3. Review Scheduling Output for Accuracy

User Story:
"As a customer, I want to review the output generated by the scheduling algorithm to ensure that the schedules meet my expectations (e.g., correct number of matches per team, no venue conflicts)."

Acceptance Criteria:

- The generated schedules are verified against the initial requirements.
- No venue or time conflicts exist in the schedules.
- Each team has the correct number of games (e.g., no team has more or fewer matches than expected).
- The product owner has confirmed the output is accurate and meets the expected criteria.

Definition of Done (DoD):

- The generated schedules are reviewed against the defined requirements.
- Any discrepancies or conflicts (venue, match count, etc.) are identified and corrected.
- The product owner reviews the schedules and confirms that they meet the defined requirements.

---

4. Gather Feedback for Iteration

User Story:
"As a customer, I want to provide feedback on the generated schedules after reviewing them, so that developers can make improvements to the scheduling algorithm in future sprints."

Acceptance Criteria:

- A feedback session is conducted with the product owner or relevant stakeholders.
- Feedback is collected regarding the accuracy, efficiency, and usability of the schedules.
- Suggestions for future improvements or issues are documented and shared with the development team.
- The product owner has acknowledged and agreed upon the feedback for future iterations.

Definition of Done (DoD):

- A feedback session is completed with stakeholders (e.g., product owner).
- All feedback is documented clearly.
- The feedback is reviewed by the team, and relevant action items are created for future sprints.

---

5. Set Up Development Environment and Version Control

User Story:
"As a developer, I want to set up the development environment and version control (Git), so that I can efficiently track changes and collaborate on the scheduling algorithm development."

Acceptance Criteria:

- The development environment is set up, including any necessary dependencies.
- Version control (Git) is initialized, and all team members have access to the repository.
- The initial project is committed to the repository.
- The environment setup is tested to ensure proper execution of the scheduling algorithm.

Definition of Done (DoD):

- The development environment is fully configured and ready for coding.
- A Git repository is created, and the initial codebase is pushed to it.
- All team members can access the project and have their development environments working.
- The product owner confirms the environment setup is correct and accessible.

---

6. Design Scheduling Algorithm

User Story:
"As a developer, I want to design a scheduling algorithm that can handle different league scenarios (e.g., 10-12 teams, different match days/venues), so that the system can generate valid schedules based on input data."

Acceptance Criteria:

- The design must handle different types of leagues (e.g., recreational, competitive).
- The algorithm design should ensure that teams are assigned appropriate match days and times without conflicts.
- Edge cases such as late registrations or fewer teams than expected should be considered.
- The product owner has reviewed and approved the design.

Definition of Done (DoD):

- A design document outlining the algorithm's logic and constraints is created.
- The design includes:
    - Input data structure and how it's parsed.
    - Scheduling constraints (e.g., maximum number of matches, team distribution).
    - Edge cases (e.g., fewer teams, late registrations).
- The document is reviewed and approved by the product owner.

---

7. Develop Scheduling Algorithm

User Story:
"As a developer, I want to implement the scheduling algorithm based on the design, so that it can automatically generate schedules for leagues with varying team counts and constraints."

Acceptance Criteria:

- The algorithm is implemented according to the design.
- The code is clean, well-documented, and follows coding standards.
- The algorithm is capable of generating valid schedules for at least two sample league scenarios.
- The product owner confirms that the implementation meets the requirements.

Definition of Done (DoD):

- The algorithm is implemented as per the approved design document.
- The code is properly commented and follows the team's coding conventions.
- The algorithm is tested with basic sample inputs to verify that it generates correct schedules.
- The product owner reviews and confirms that the implementation meets expectations.

---

8. Test the Algorithm with Given Test Cases

User Story:
"As a developer, I want to test the algorithm using predefined test cases (e.g., 10 teams, double-header matches, 12 teams with a late registrant) to ensure that the scheduling logic works correctly."

Acceptance Criteria:

- The algorithm passes all predefined test cases.
- For each test case, the generated schedules match the expected output.
- Any issues or discrepancies in the results are logged and fixed.
- The product owner has confirmed that the test cases cover all necessary scenarios.

Definition of Done (DoD):

- All test cases have been executed, and the results have been compared against the expected output.
- Any issues found are fixed and retested.
- The product owner confirms that the test cases are comprehensive and the algorithm works as expected.

---

9. Document the Algorithm Design and Logic

User Story:
"As a developer, I want to document the scheduling algorithm's design and logic, so that others can understand how the system generates schedules and maintain it in the future."

Acceptance Criteria:

- A detailed document explaining the algorithm’s design and logic is created.
- The document includes:
    - High-level description of the algorithm.
    - Step-by-step explanation of how the schedule is generated.
    - Considerations for edge cases and error handling.
- The product owner has reviewed and approved the documentation.

Definition of Done (DoD):

- The algorithm's design and logic are thoroughly documented.
- The document includes all relevant sections (e.g., algorithm flow, input/output, edge cases).
- The product owner reviews and approves the documentation.

---

10. Optimize Algorithm Efficiency

User Story:
"As a developer, I want to review and optimize the efficiency of the scheduling algorithm, so that it can handle larger leagues and generate schedules faster."

Acceptance Criteria:

- The algorithm has been profiled to identify performance bottlenecks.
- Any slow or inefficient parts of the algorithm are optimized for better performance.
- The algorithm is retested to ensure correctness after optimization.
- The product owner confirms that the optimizations improve performance without breaking functionality.

Definition of Done (DoD):

- The algorithm is optimized for better performance based on profiling results.
- The optimization does not affect the correctness of the algorithm.
- The product owner reviews the optimized version and confirms it meets performance expectations.

---

11. Document the Installation and Running Process in README

User Story:
"As a developer, I want to document how to install and run the scheduling algorithm on a local machine, so that users can easily set up and test the system."

Acceptance Criteria:

- The README file includes installation instructions (e.g., required libraries or dependencies).
- The README includes clear steps to run the algorithm on a local machine.
- Instructions for running test cases and verifying algorithm functionality are provided.
- The product owner confirms the instructions are clear and easy to follow.

Definition of Done (DoD):

- The README file has been updated with installation and usage instructions.
- The instructions are clear and comprehensive.
- The product owner reviews and approves the README.