ROBERT GORDON UNIVERSITY ABERDEEN

INFORMATICS INSTITUTE OF TECHNOLOGY

BSc (Hons) Artificial Intelligence and Data Science

Module: CM1603 Database Systems

Individual Coursework Report

Module Leader: Ms. Malsha Fernando

RGU Student ID : 2425472

IIT Student ID : 20240892

Student Name : Lidiya Rajapakse

Table of Contents

Table of Figures

1. INTRODUCTION

PROBLEM IDENTIFICATION

University gaming clubs often need to form balanced teams quickly for tournaments, friendly matches and inter-university events. Manual team formation is time consuming and error-prone.organizers must balance game/sport preferences, skill levels, playing roles, and personality dynamics while keeping teams fair and competitive. For large membership lists, doing this manually causes uneven teams, dissatisfied members, and extra administrative overhead.

PROPOSED SOLUTION

TeamMate runs as a Java application. It automates the whole process of forming teams based on data from member surveys. The app gathers quick surveys that cover things like personality scores, interests, role preferences, and skill levels. Then it sorts out personality types from that info. After that, it kicks off a matching algorithm to build teams of size N. Those teams aim to boost diversity in interests. They also make sure roles get covered properly. Plus, the mix of personality types helps create stronger team dynamics overall. The system handles CSV import and export without issues. It supports concurrency to deal with big datasets. And it includes solid error handling to keep things running smooth.

PROJECT SCOPE

- Ingestion of survey data (CSV) with basic validation.
- Classification of personality based on ranges of scores: Leader, Balanced, Thinker.
- The team formation algorithm that considers game interest, role coverage, skill-level balancing, and personality mix.
- Files: load sample CSV, write formed teams CSV.
- Exception and concurrency handling: threads for survey processing, parallel team formation.
- Unit and integration tests, concurrency tests, file integrity checks.
- Deliverables: Runnable Java Application (.zip), Full Report (.pdf), and Version Control Logs.

TIMELINE

Week	Description
Week 1	Requirement analysis, actors, use cases, and initial

Design kickoff	UML (use case, class diagram).
Week 2 Detailed design	Activity & sequence diagrams; finalize class responsibilities (Participant, Team, TeamBuilder, PersonalityClassifier, CSVUtil, Validator)
Week 3-4 Implementation phase 1	Core classes, CSV load/save, PersonalityClassifier, Participant model, validations.
Week 5 Implementation phase 2	Matching algorithm, role/interests balancing, concurrency integration , thread pools for processing & team formation.
Week 6 Integration & UI	Integrate modules, add console, logging, exception handling.
Week 7 Testing	Unit tests, integration tests, concurrency tests, file checks, prepare report.
Week 8 Buffer & polish	Resolve issues discovered in tests, create screenshots and version control logs, conclude report and package submission.

2. REQUIREMENT ANALYSIS

STAKEHOLDERS

- Organizer (Primary actor)

Role: Uploads CSV, defines team size N, initiates team formation, and reviews/exports teams.

Requirements: Form team quickly and reliably, control over the constraints, human-readable output.

- Participant (Secondary actor)

Role: Fills out survey: personality questions, role preference, interests, and skill level.

Requirements: Correct representation of preferences, equitable assignment of teams.

KEY FUNCTIONAL REQUIREMENTS

Functional Requirement ID	Functional Requirement	Description
FR1	CSV Import	System must load participant data from a given CSV (starter pack). It should skip/flag malformed rows and continue processing others.
FR2	Process Survey Answers	Process survey answers to compute a personality score for each participant - sum of 5 questions - and map to personality type: Leader/Balanced/Thinker.
FR3	Role & Interest Capture	Store participant preferred role, interest(s), and skill level. Support multiple interests if present.
FR4	Team Formation	For any given team size N, teams are to be formed such that • Diverse interests across teams where possible. • Role coverage • Mix of personality types per team. • Balanced average of skill.
FR5	CSV Export	Save final teams to formed_teams.csv
FR6	Error Reporting	Reduce errors by providing appropriate error messages and logs regarding invalid input, missing fields, file I/O errors.
FR7	Concurrency	The program will use threads to parallelize survey processing and the formation of teams for large input sizes.
FR8	Tests	Include unit tests for key algorithmic pieces and integration tests for CSV I/O and team formation.

NON-KEY FUNCTIONAL REQUIREMENTS

Non-Functional Requirement ID	Non-Functional Requirement	Description

NFR1	Robustness	The system should not crash if there are invalid/incomplete CSV entries.
NFR2	performance	For normal club sizes of up to several hundred participants, the formation of teams should be completed within reasonable time by using concurrency.
NFR3	Maintainability	Code should be written in a clean OOP design, well documented, and with modular classes to allow easy extension.
NFR4	Portability	The application shall run on any desktop with Java and minimal external dependencies.
NFR5	Traceability	Provide logs of producing the report and version control screenshots; add timestamps.
NFR6	Usability	Console prompts and output .csv format should be clear and well organized for the organizers.

USE CASE DIAGRAM

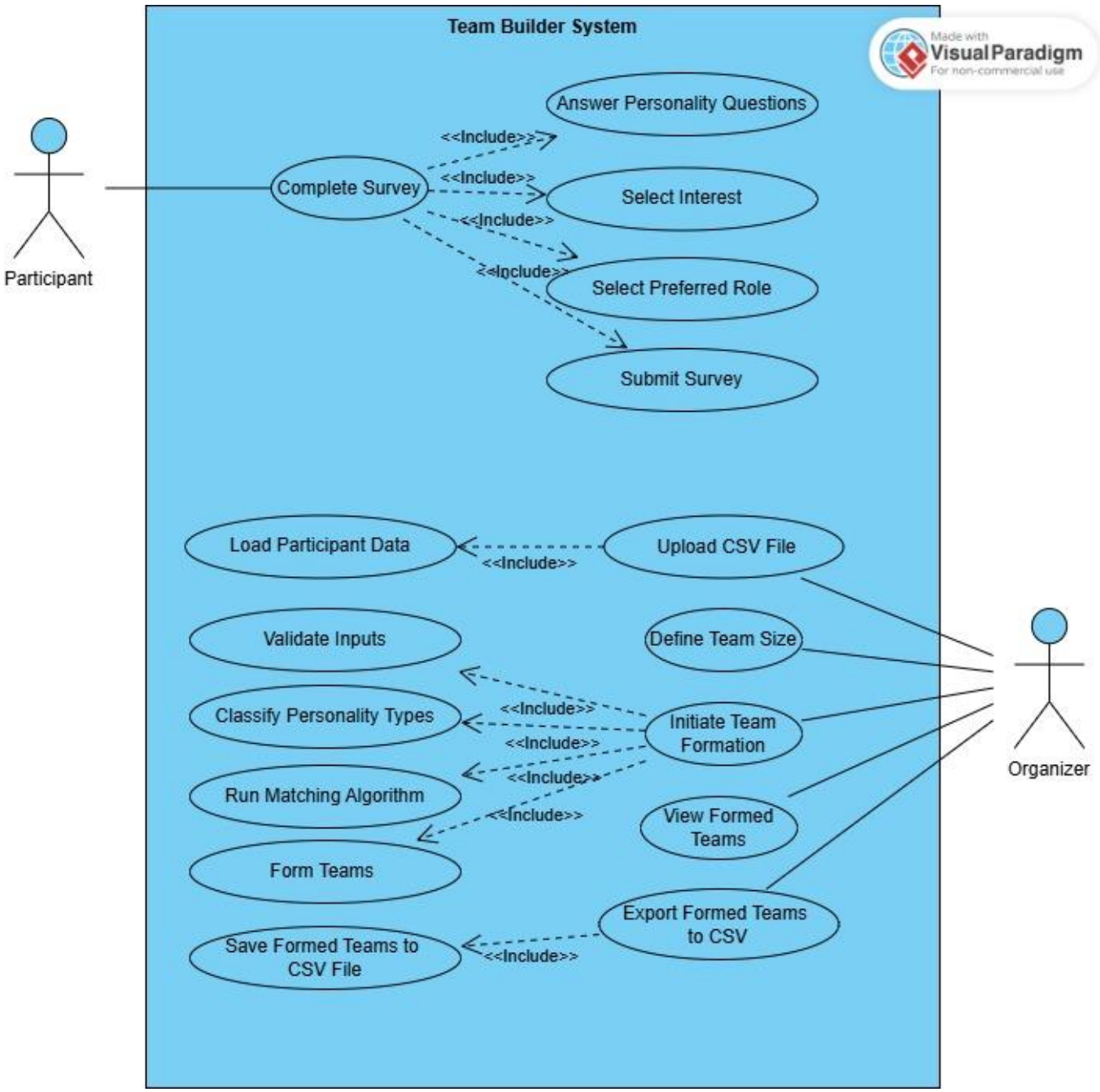

Figure 1:Use case diagram

USE CASE DESCRIPTIONS

1. Complete Survey

Use Case Name:	Complete Survey
Section	Main
Actor	Participant
Overview:	Describes how the participant completes the survey by answering personality questions, selecting interests, and choosing a preferred role.
Purpose:	To collect all required participant data for personality classification and team formation.
Preconditions:	• The system is running. • Participant has access to the survey interface.
Postconditions:	• Survey responses are stored in memory or CSV.
Main Flow:	
1. Participant selects the “Complete Survey” option.	
2. System displays the survey form.	
3. Participant answers personality questions.	
4. Participant selects game/sport interest.	

5. Participant selects a preferred role.	
6. System validates all inputs.	
7. System saves the survey responses.	
Alternative Flow:	• If any input is invalid or missing, the system prompts the participant to correct it. • If saving fails, the system displays an error.
Includes:	• Answer Personality Questions • Select Interest • Select Preferred Role • Submit Survey
Extends:	None

2. Upload CSV

Use Case Name:	Upload CSV
Section	Main
Actor	Organizer
Overview:	Describes how the organizer uploads a CSV file containing sample participant data.
Purpose:	To import existing participant data into the system.

Preconditions:	• Organizer has a valid CSV file. • System can access storage.
Postconditions:	• CSV data is loaded into the system.
Main Flow:	
1. Organizer selects the “Upload CSV” option.	
2. System asks for file path.	
3. Organizer provides the CSV file.	
4. System reads and loads participant data.	
Alternative Flow:	• If the file is missing, an error is displayed. • If CSV format is invalid, system rejects the file.
Includes:	• Load Participant Data
Extends:	None

3. Define Team Size

Use Case Name:	Define Team Size
Section	Main

Actor	Organizer
Overview:	Organizer specifies the desired team size (3–6).
Purpose:	To guide the team formation algorithm.
Preconditions:	• Participants are loaded.
Postconditions:	• Valid team size is saved.
Main Flow:	
1. Organizer enters team size.	
2. System validates it.	
3. System saves the team size.	
Alternative Flow:	• If size is outside 3–6, system rejects input.
Includes:	None
Extends:	None

4. Initiate Team Formation

Use Case Name:	Initiate Team Formation
Section	Main
Actor	Organizer
Overview:	Organizer commands the system to begin forming balanced teams.
Purpose:	To generate teams according to matching rules.
Preconditions:	• Participant data loaded. • Team size defined.
Postconditions:	• Balanced teams are created.
Main Flow:	
1. Organizer selects “Form Teams”.	
2. System validates all required data.	
3. System classifies personality types.	
4. System runs the matching algorithm.	
5. System forms balanced teams.	

6. Teams become ready for viewing/exporting.	
Alternative Flow:	• If validation fails, system shows an error. • If team formation is not possible, system alerts the organizer.
Includes:	• Validate Inputs • Classify Personality Types • Run Matching Algorithm • Form Teams
Extends:	None

5. View Formed Teams

Use Case Name:	View Formed Teams
Section	Main
Actor	Organizer
Overview:	Allows organizer to view the generated teams.
Purpose:	To review the results of team formation.
Preconditions:	• Teams have been formed.
Postconditions:	• Teams are displayed on screen.

Main Flow:	
1. Organizer selects “View Teams”.	
2. System shows all formed teams.	
Alternative Flow:	None
Includes:	None
Extends:	None

6. Export Formed Teams to CSV

Use Case Name:	Export Formed Teams to CSV
Section	Main
Actor	Organizer
Overview:	Organizer exports the formed teams to a CSV file.
Purpose:	To store team formation results externally.
Preconditions:	• Teams exist.
Postconditions:	• A CSV file is created containing all teams.

Main Flow:	
1. Organizer selects “Export Teams”.	
2. System writes data to a CSV file.	
3. System confirms successful export.	
Alternative Flow:	• If file access fails, system shows an error.
Includes:	• Save Formed Teams to CSV
Extends:	None

3. ACTIVITY DIAGRAMS

1. Complete CSV

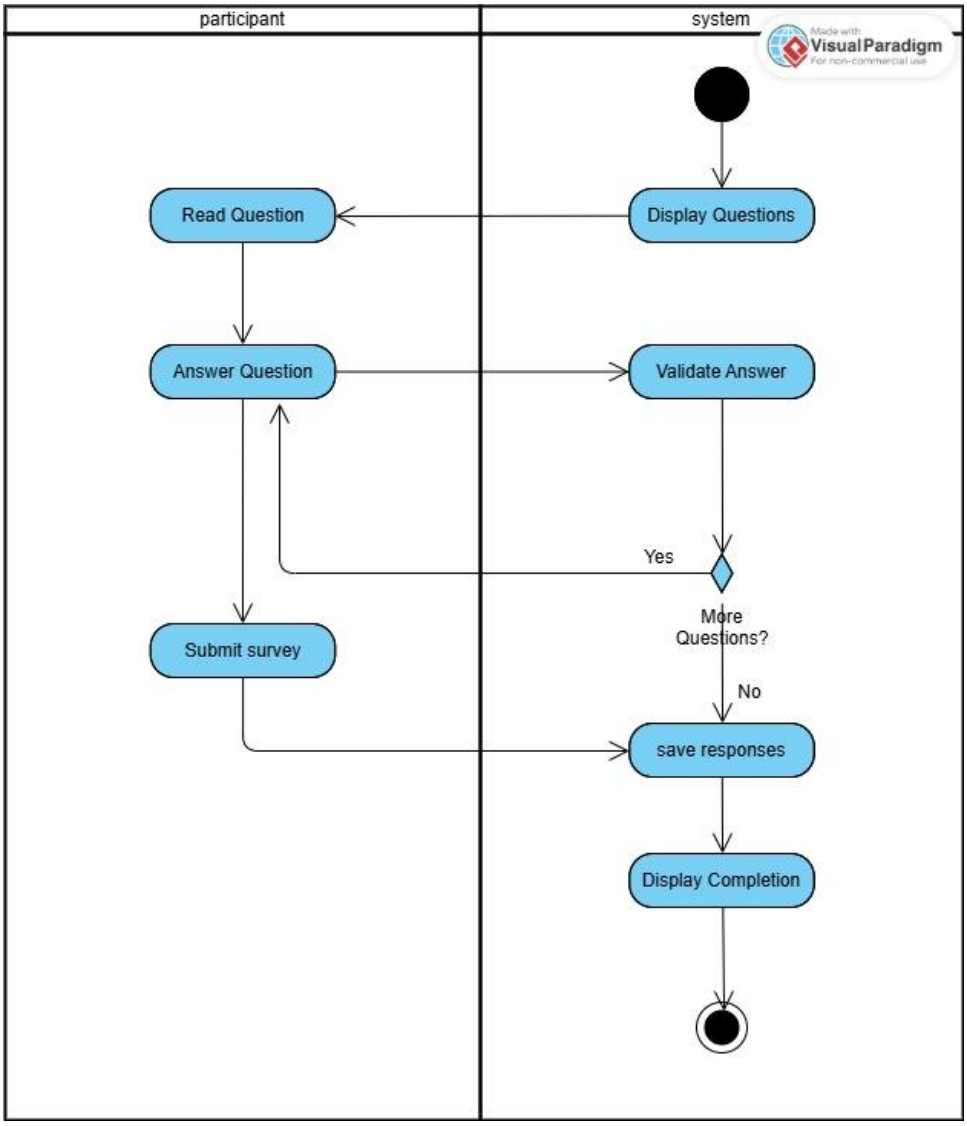

Figure 2:Upload CSV activity diagram

2. Upload CSV

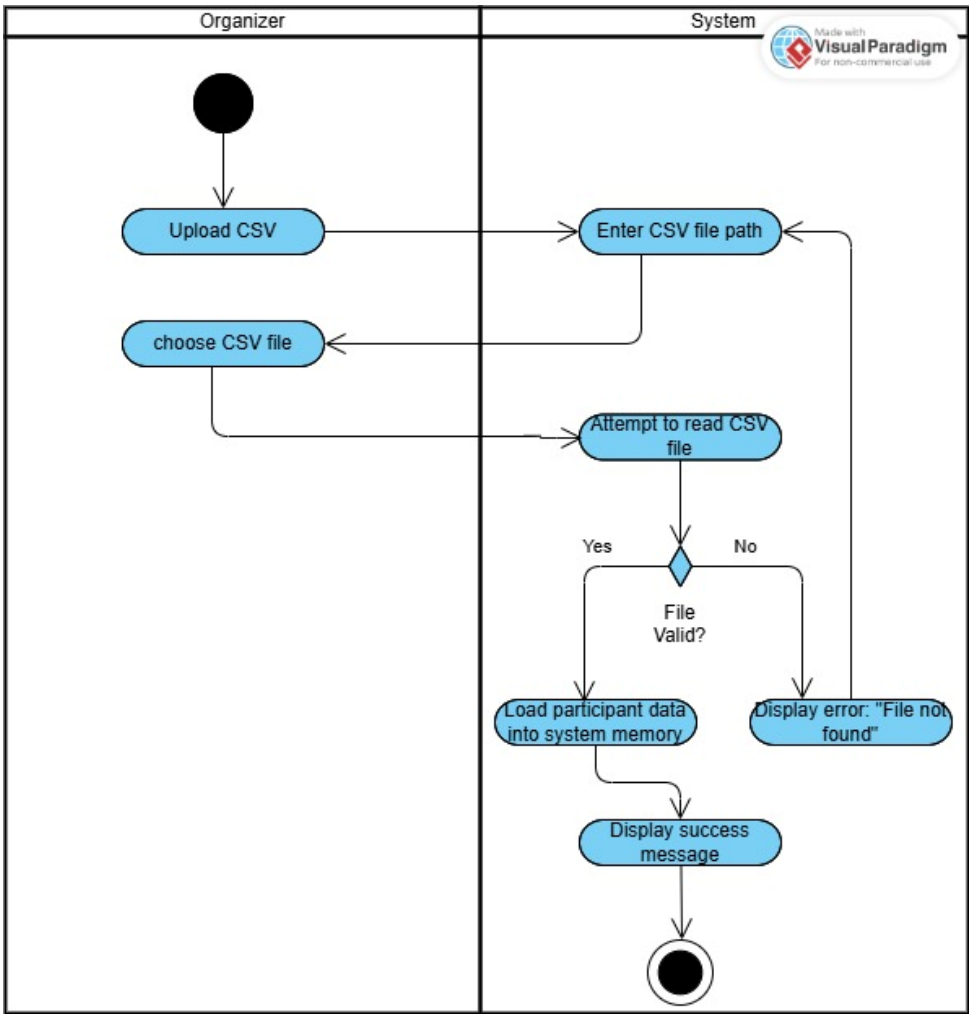

Figure 3:Upload CSV activity diagram

3. Define team size

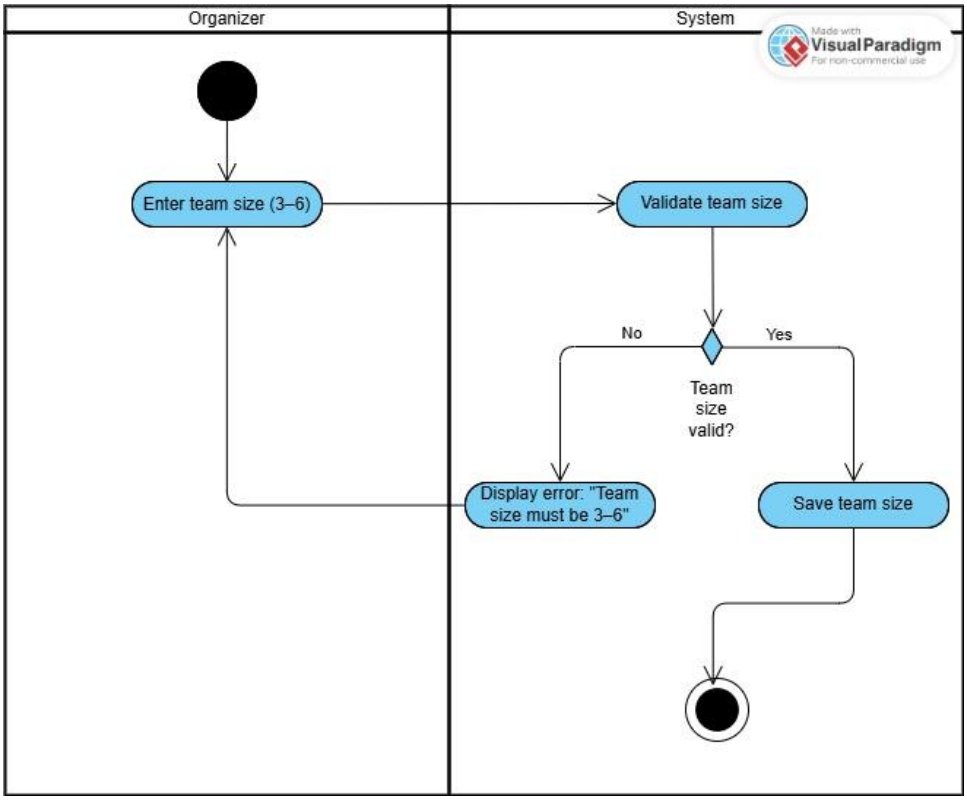

Figure 4:Define team size activity diagram

4. Initiate team formation

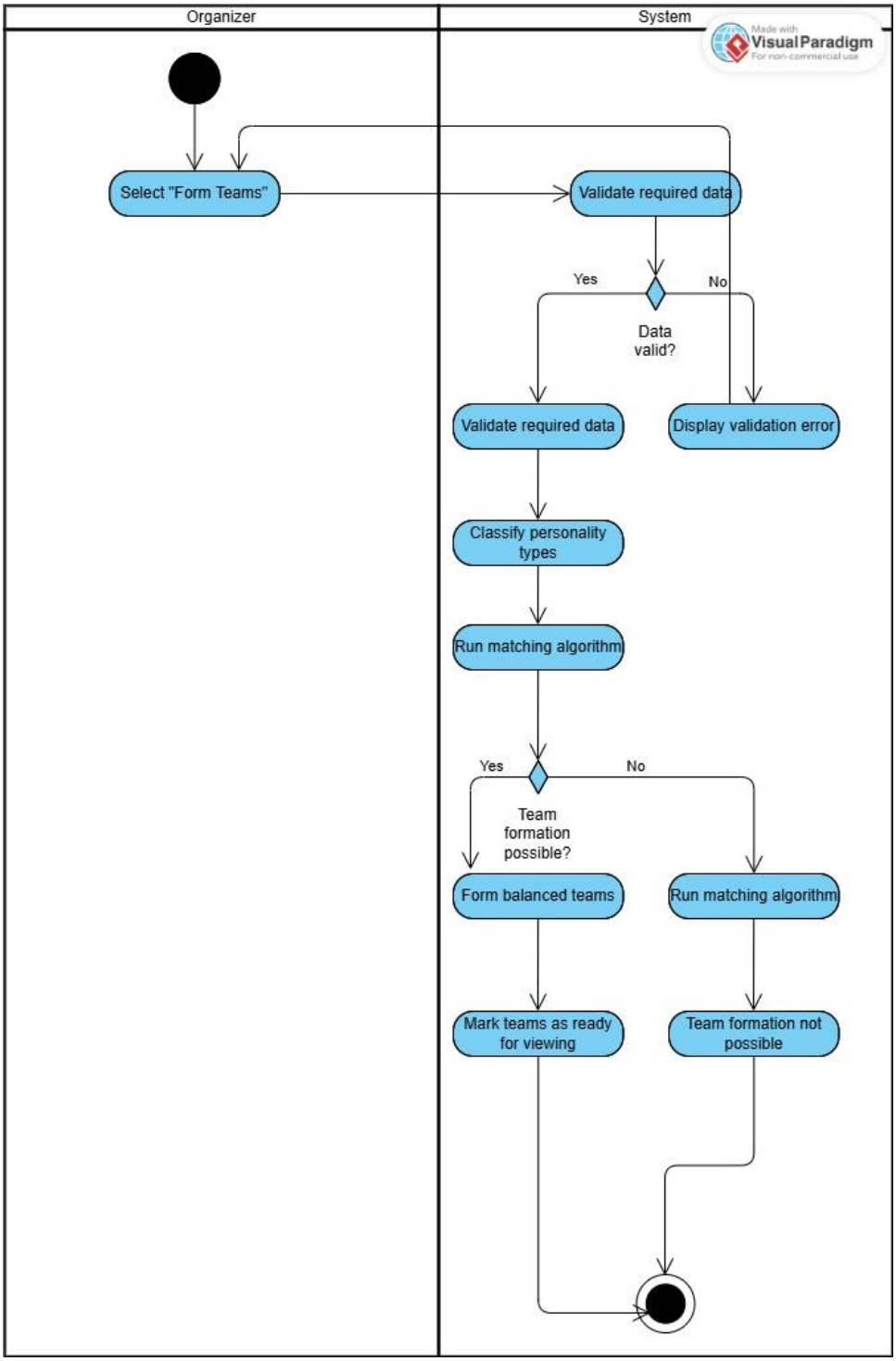

Figure 5:Initiate team formation activity diagram

5. View formed teams

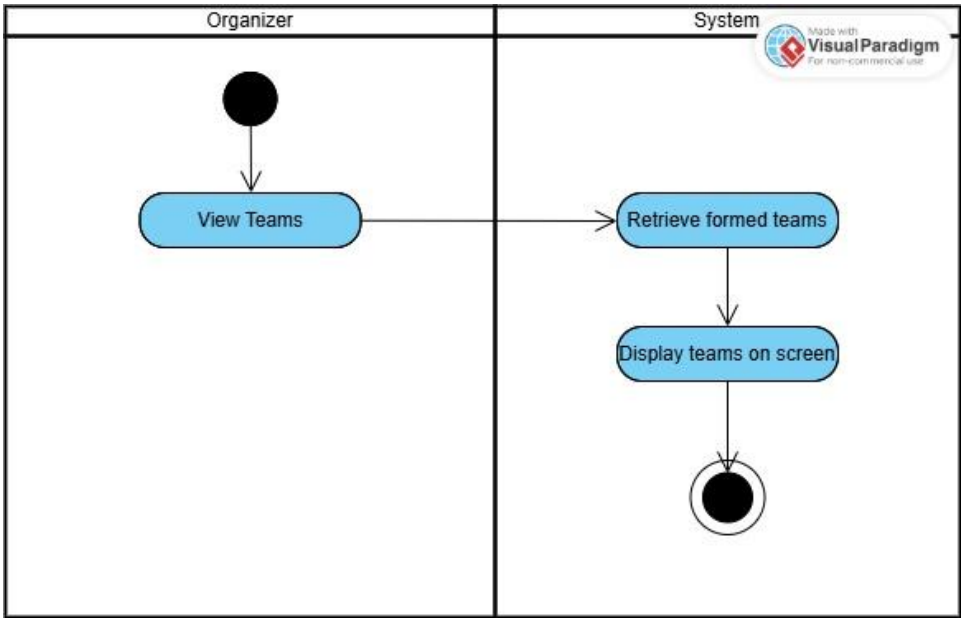

Figure 6:View formed teams activity diagram

6. Export formed teams to CSV

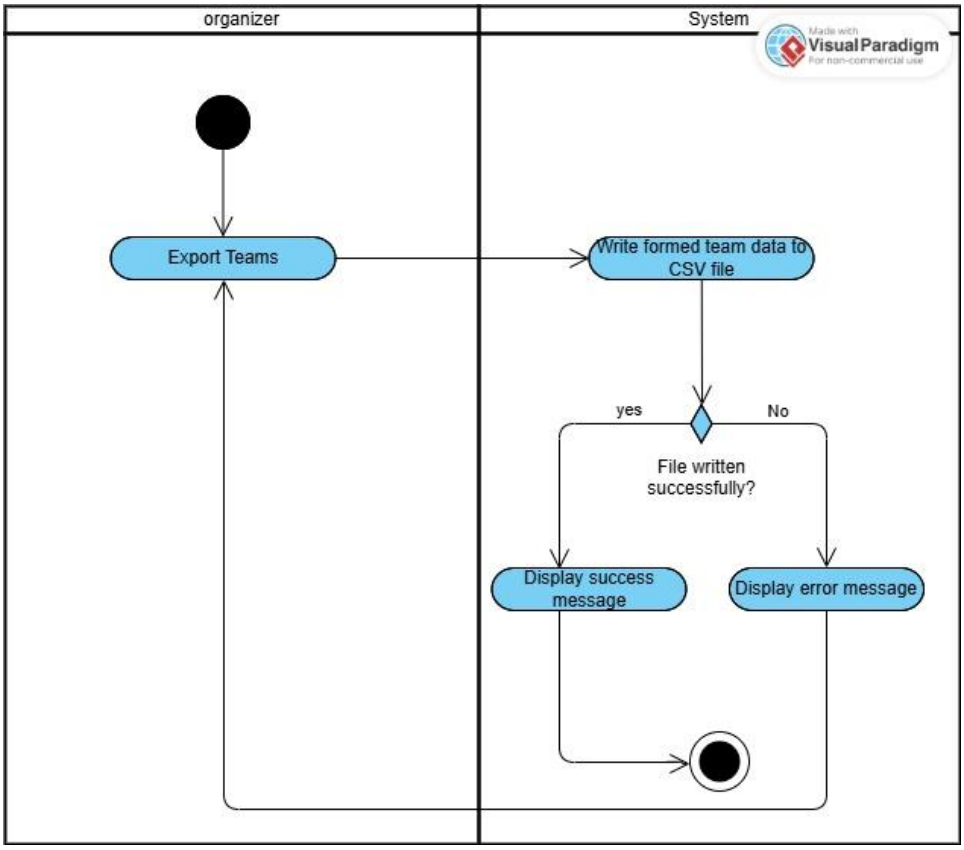

Figure 7:Export formed teams to CSV

4. A SINGLE CLASS DIAGRAM

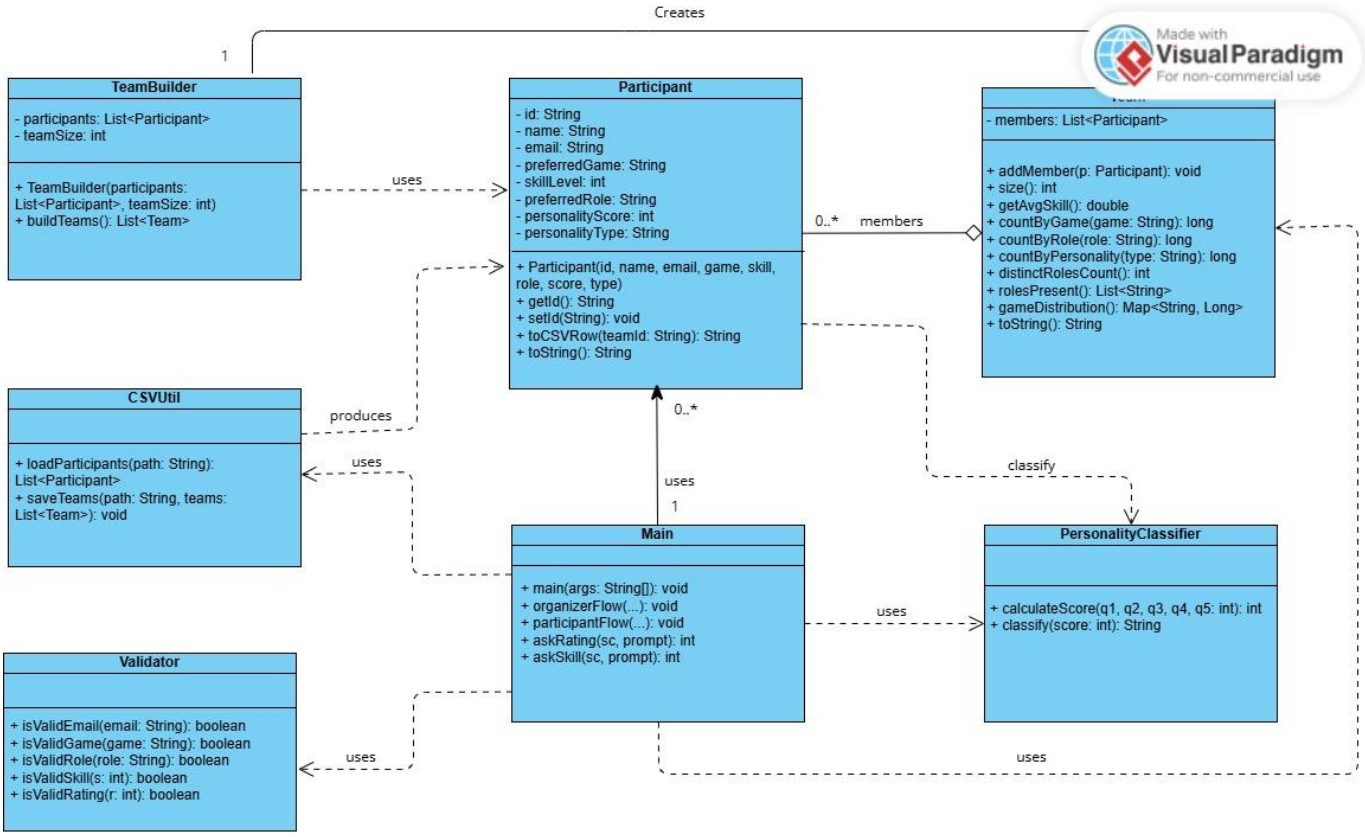

Figure 8:Class diagram

5. SEQUENCE DIAGRAMS

1.Complete survey

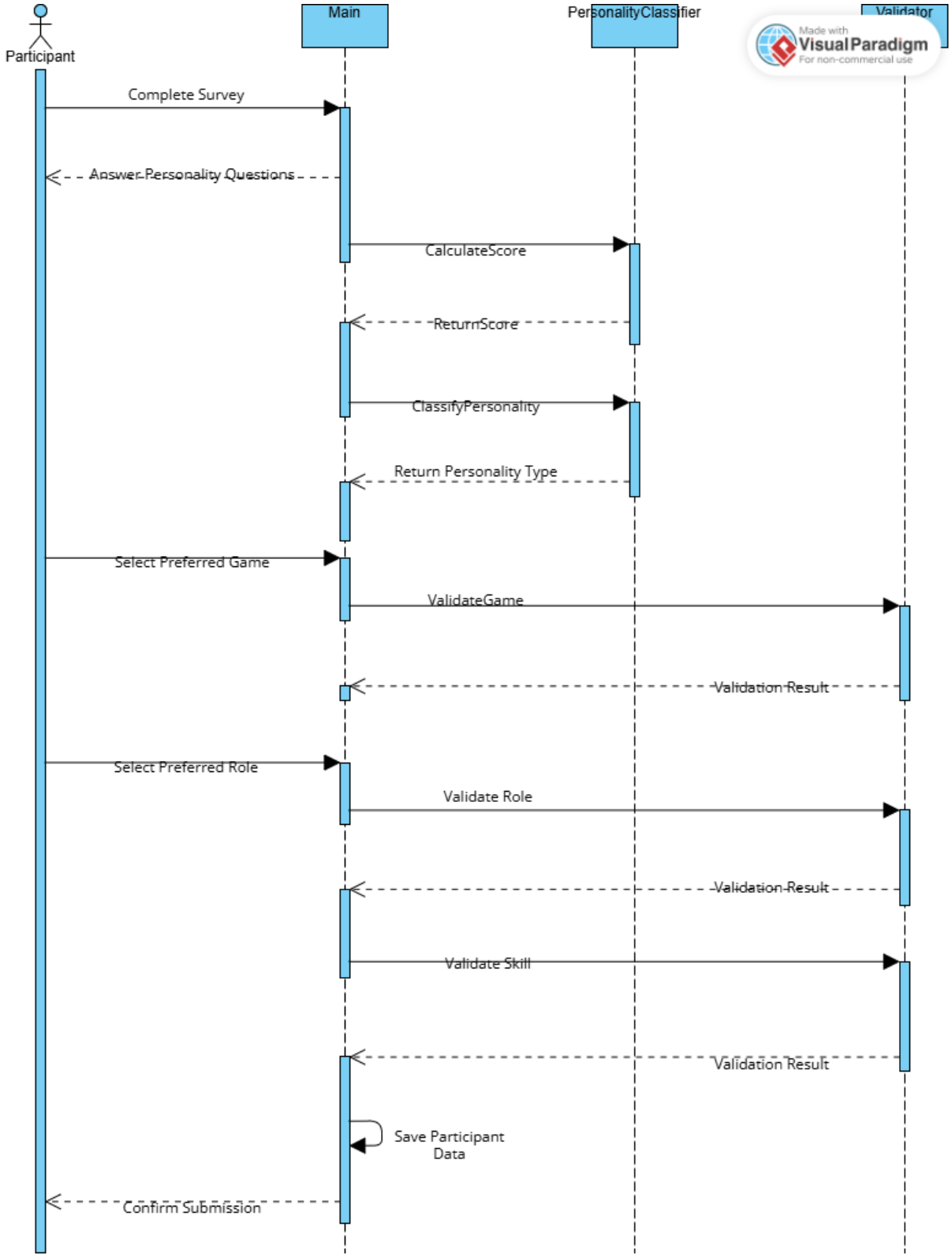

Figure 9:Complete survey sequence diagram

2.Upload CSV

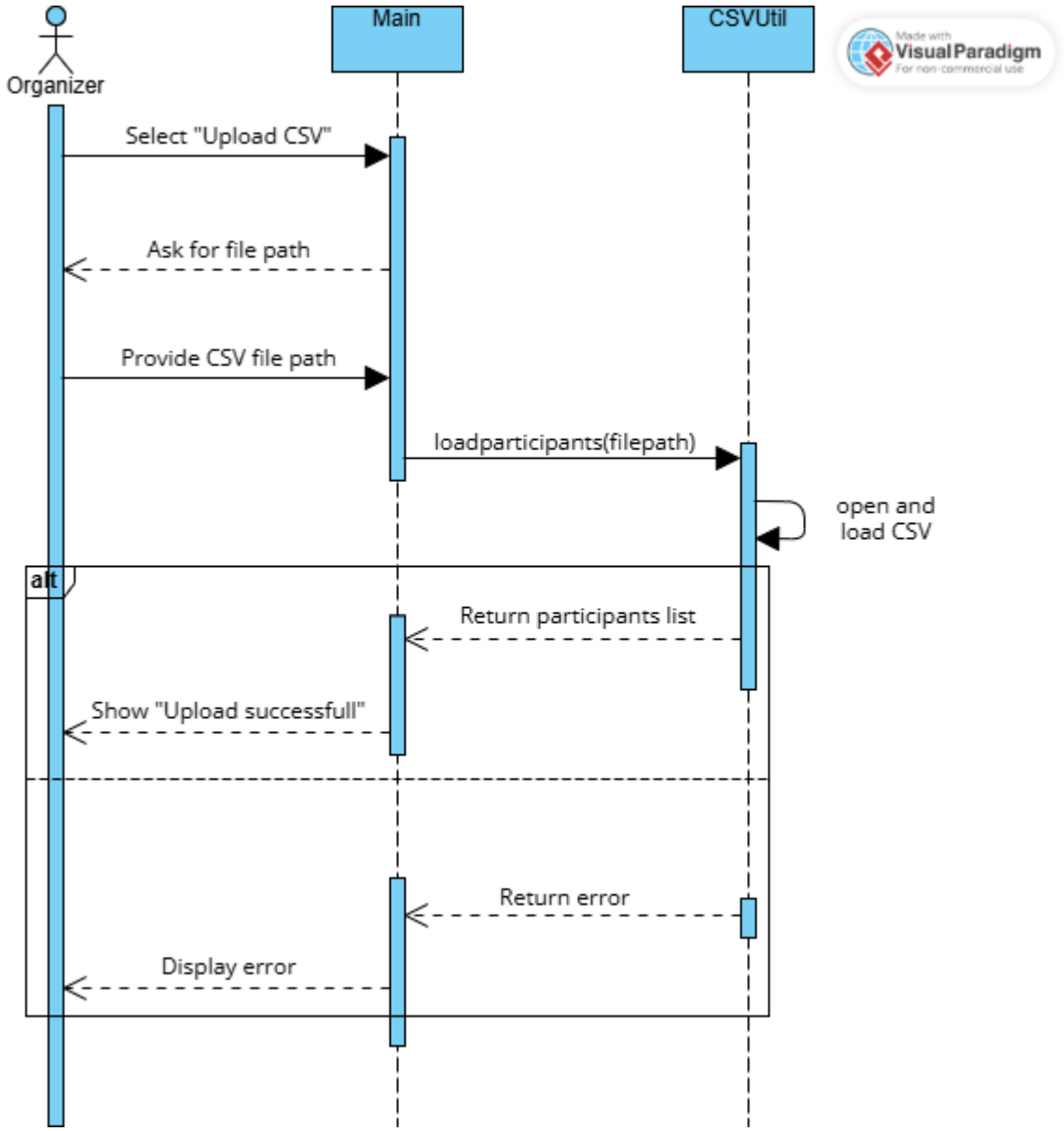

Figure 10:Upload CSV sequence diagram

3.Define team size

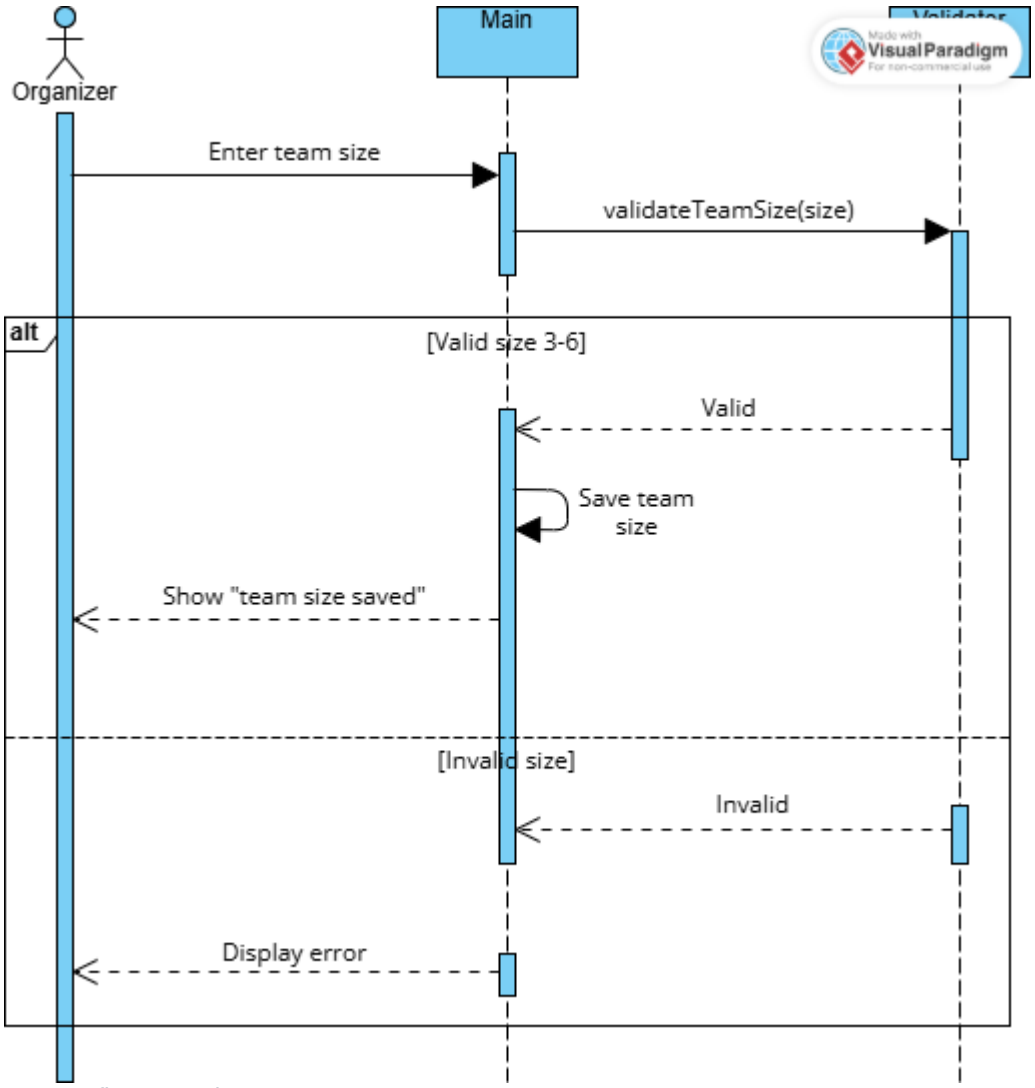

Figure 11:Define team size

4.Initiate team formation

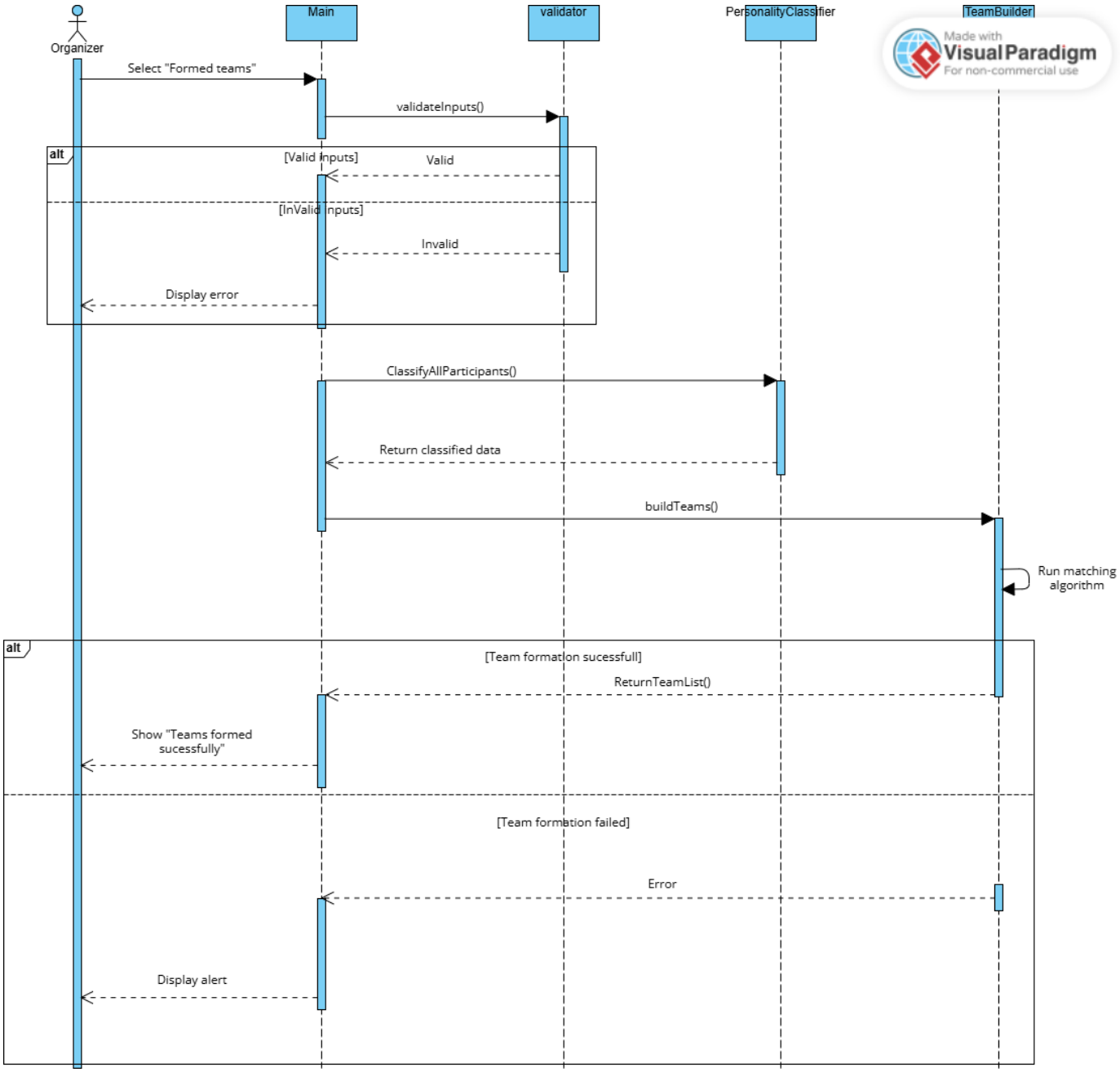

Figure 12:Initiate team formation sequence diagram

5.View formed teams

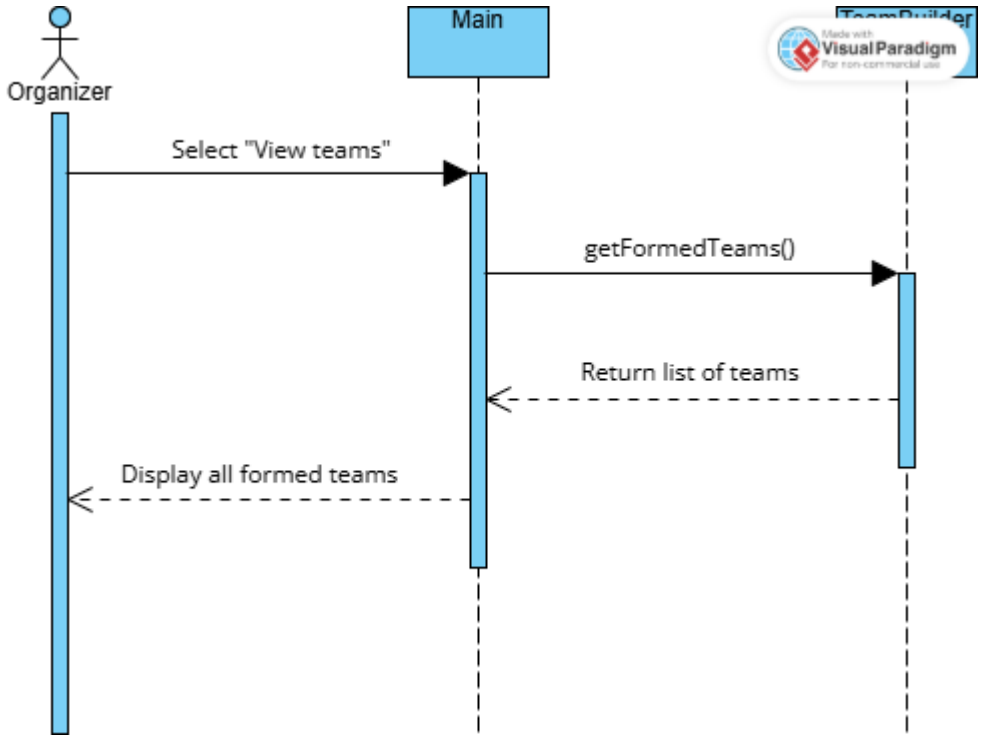

Figure 13:view formed teams sequence diagram

6.Export formed teams to CSV

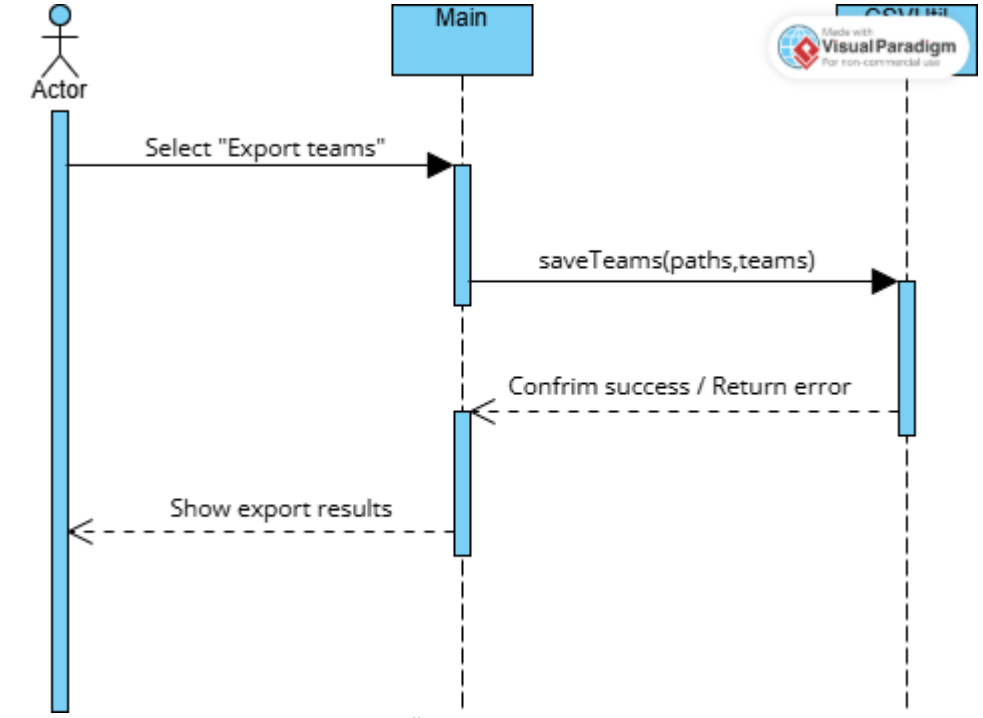

Figure 14:Export teams to CSV sequence diagram

6. TESTING AND EVALUATION

TESTING

- **Unit testing:** Testing he smallest units -PersonalityClassifier, Validator, CSV parsing logic, role/interest parsing.
- **Integration testing:** Combined components - CSV import -> classification -> team formation -> CSV export.
- **Concurrency testing:** Stress tests with survey processing and team formation running in parallel alongside verifying thread safety and consistent outputs.
- **System testing:** End-to-end, running the full pipeline with sample CSVs, edge cases, and large synthetic datasets.
- **User Acceptance Testing**: Test organizer scenarios.
- **File integrity tests:** Check that input/ output CSVs maintain expected formatting and that no data loss occurs.

DESK CHECKING

Step	Component / Action	Input / Operation	Expected Output / Observation	Status
1	Participant Survey	Participant enters: - • ID: P101 • Name: Alice • Email: lidiya@gmail.com • Game: FIFA • Role: Attacker • Ratings: Q1=5, Q2=4, Q3=3, Q4=4, Q5=5 • Skill: 7	System captures all inputs and stores temporarily	Pass
2	Input Validation	Validate ratings, skill, game, role	Ratings 1–5, Skill 1–10, Game & Role valid	Pass
3	Personality Classificatio	Calculate score: 5+4+3+4+5=21 →21×4=84	Personality Type = Balanced	Pass

	n			
4	Team Formation	Assign participant to a team ensuring role diversity and personality mix	Participant added to Team 1	Pass
5	CSV Export	Save participant/team data to CSV	Row: *Team1,P101,lidiya@gmail.com,FIFA,Attacker,7,84,Balanced*	Pass
6	Error Handling	Participant enters invalid skill: 15	System prompts: "Skill must be between 1–10"	Pass
7	Concurrency Simulation	Multiple participants processed concurrently	No duplication or data loss,each participant assigned once	Pass

EVALUATION

- Correctness: Teams satisfy functional constraints for most of the realistic inputs.
- Robustness: Application handles malformed inputs logs meaningful messages.
- Performance: Concurrency increases the throughput against single threaded runs for large data sets.
- Usability: CSV output is clear and can be used by organisers.
- Documentation & tests: Report includes design diagrams, test evidence, and version control logs.

7. REFERENCES

Apache Commons. (n.d.) *Apache Commons CSV User Guide*. Available at: https://commons.apache.org/proper/commons-csv/ (Accessed: 12 November 2025).

Fowler, M. (2010) *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd edition. Boston: Addison-Wesley.

Goetz, B., Peierls, T., Bloch, J., Bowbeer, J., Holmes, D. and Lea, D. (2006) *Java Concurrency in Practice*. Boston: Addison-Wesley.

JUnit Team. (2025) *JUnit 5 User Guide*. Available at: https://junit.org/junit5/docs/current/user-guide/ (Accessed: 20 November 2025).

Pro Git Book. (2019) *Pro Git*. 2nd edition. Available at: https://git-scm.com/book/en/v2 (Accessed: 20 November 2025).

RGU Starter Pack. (2023) *TeamMate Project Starter Pack*. Available from: [C:\Users\USER\Documents\IIT\Stage 2\Semester 1\CM2601-Object oriented development\Assesment] (Accessed: 12 November 2025).

8. APPENDIX

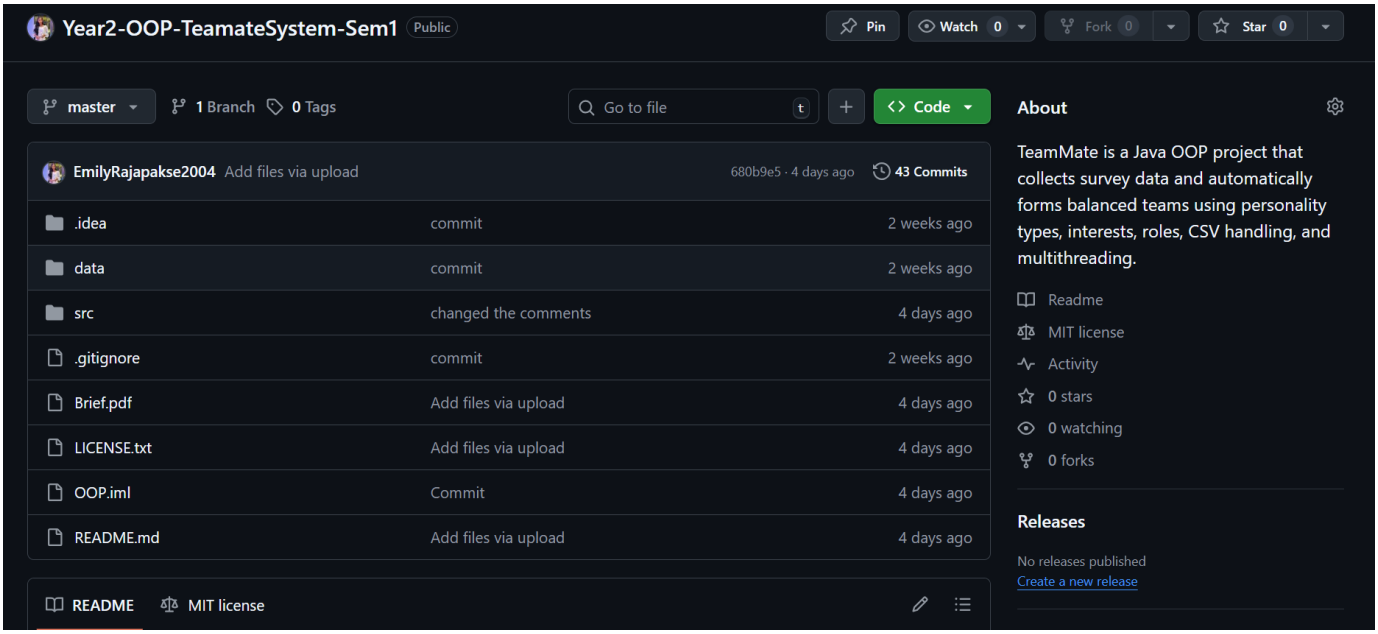

Figure 15:Github repository

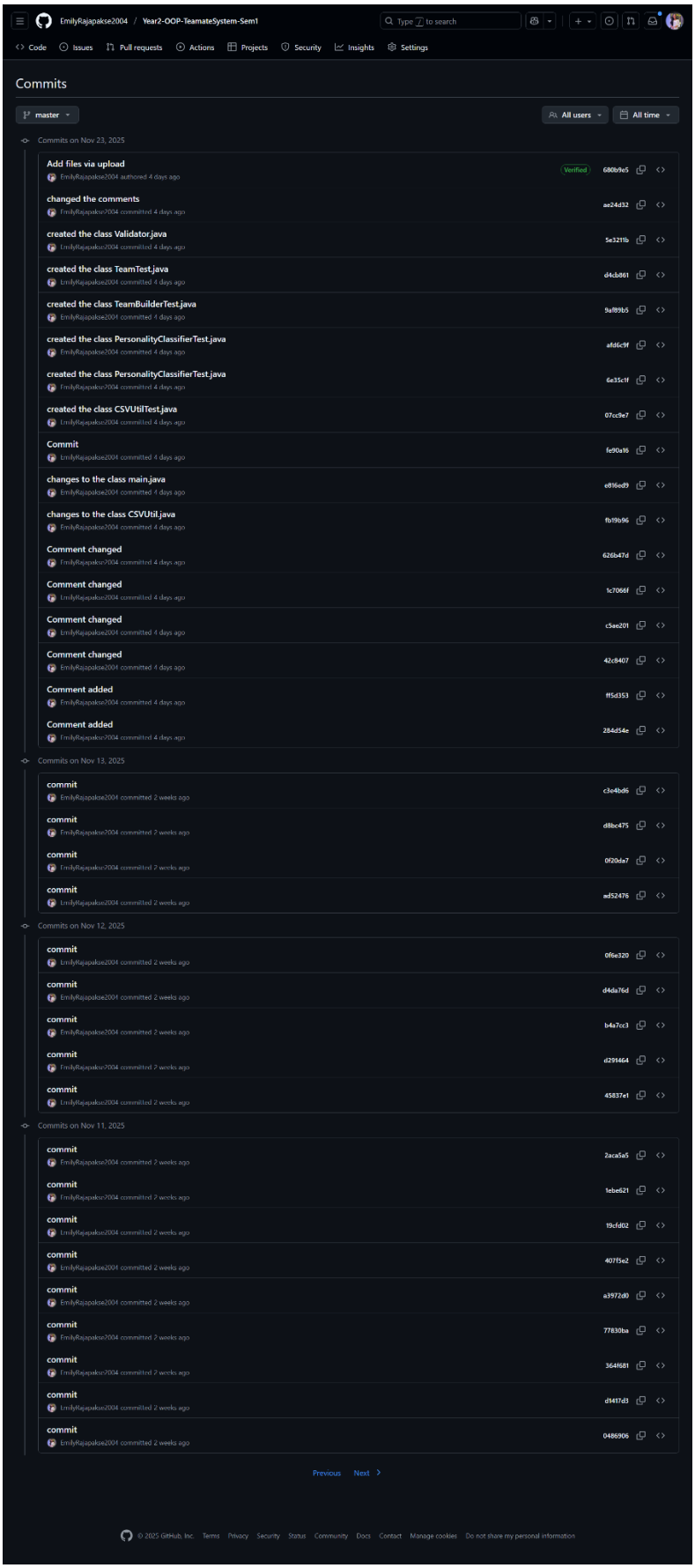

Figure 16:Commit history