PROJECT STORM FUNCTIONAL REQUIREMENTS SPECIFICATION TEAM NAME

Renaldo van Dyk 12204359
Sean Hill 12221458
Shaun Meintjes 133310896
Johann Dian Marx 12105202

Client:

Linda Marshall & Vreda Pieterse
lmarshall@cs.up.ac.za,vpieterse@cs.up.ac.za

GitHub link

Contents

1 Vision and Scope

1.1 History and Background

1.1.1 A Brief History of Project STORM

In 2013 lecturers of the department of Computer Science, which resides at the University of Pretoria, approached honors students of the module "Educational Software Development" to develop a team shuffling system. The system was to be used by the lecturers of the module "Software Engineering" to determine teams for the "Rocking the boat" exercise of the "Software Engineering" module using a set of lecturer defined criteria to create the teams with. An incomplete requirements specification document was designed and the project was halted.

1.1.2 Project Background

The lecturers of the "Software Engineering" module sought the need for such a shuffling tool, this time approaching students of the "Software Development" module. The requirements specification previously developed will be used as a starting point. This document will be stripped down to a "basic system" requirements specification, not completely discarding functionality specified in the previous documentation but adding relevant functionality as the development life-cycle persists.

1.2 Project Scope

The complete system should enable users to build teams, from a list of subjects, by selecting a set of criteria. This will aid the users in such a way that the users do not have to build the teams manually, which may require a lot of time and effort. The users can spend their time rather on analyzing the results of each "Rocking the Boat" round to change the criteria for the next round more effectively.

2 Application requirements and design

2.1 Modular Design

The system is to be a modular system which allows for:

- Only a subset of modules to be deployed. Minimally the system will require the core modules to be deployed.
- Further modules to be added at a later stage.

To this end there should be:

- Minimal dependencies between modules, and
- No dependencies of core modules on any add-on modules.

Modular design allows that each module encapsulates information that is not available to the rest of the program. This information hiding reduces the cost of subsequent design changes when future functionality is added to STORM. For example, if at a later stage functionality is added to allow for personality tests to be completed within STORM and results are automatically pulled in, a new module can be added without affecting other modules.

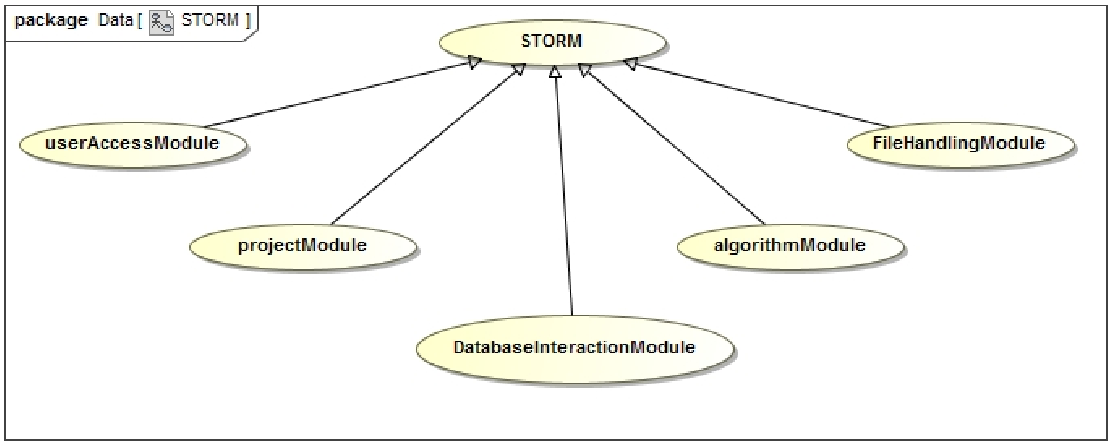

Figure 1: High level overview of STORM

2.2 User Access Module

This module deals with the STORM user access, specifically signing up, logging in, logging out and user authentication.

2.2.1 Use-cases

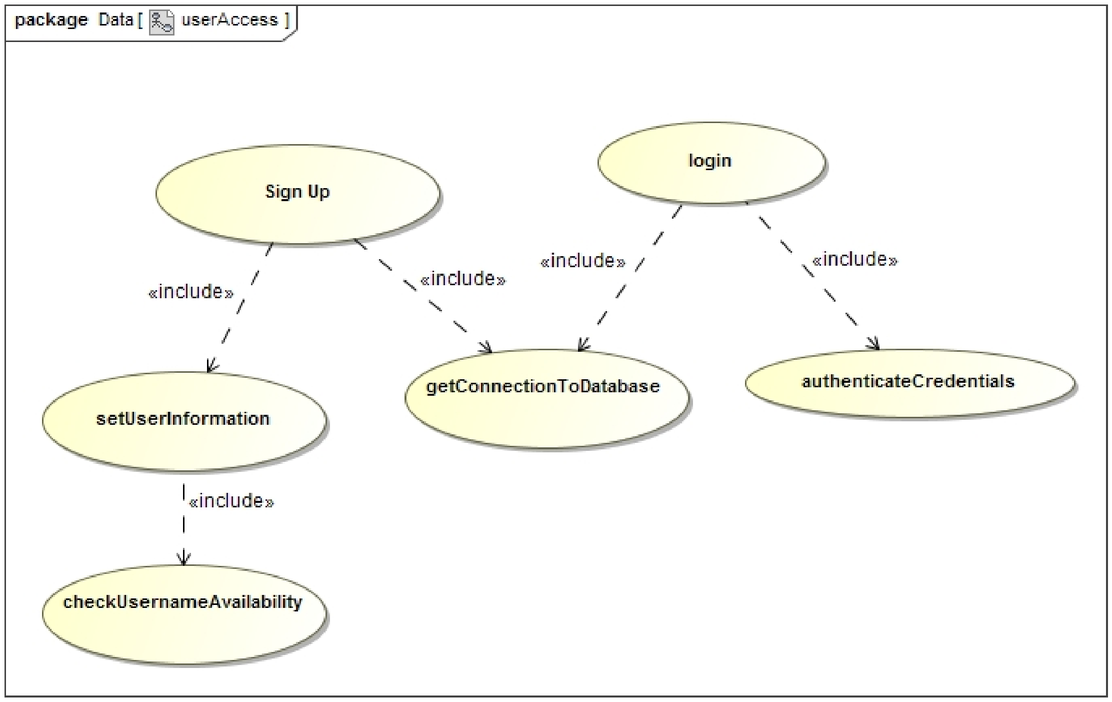

Figure 2: User access module use case

1. Sign Up

   Priority: Critical.

   Pre-condition: Client must have a valid e-mail address.

   Post-condition: Client has a STORM profile.

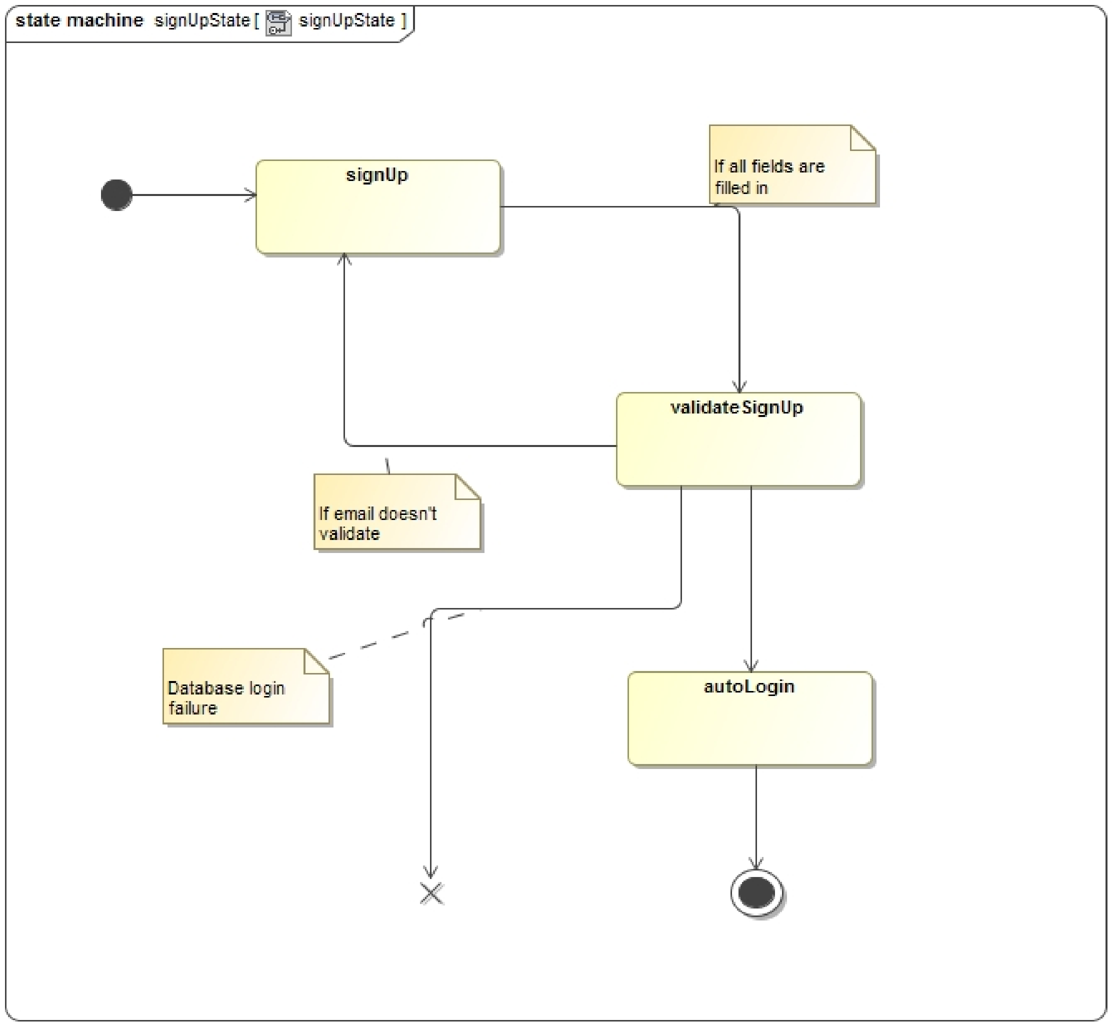
state machine signUpState [signUpState]
signUp
If all fields are filled in
validateSignUp
If email doesn't validate
Database login failure
autoLogin

Figure 3: SignUp state diagram

2. Login

   Priority: Critical.

   Pre-condition: Client must have a STORM profile.

   Post-condition: Client can now use STORM functionality.

3. Log out

   Priority: Critical.

Pre-condition: Client must be logged into STORM.

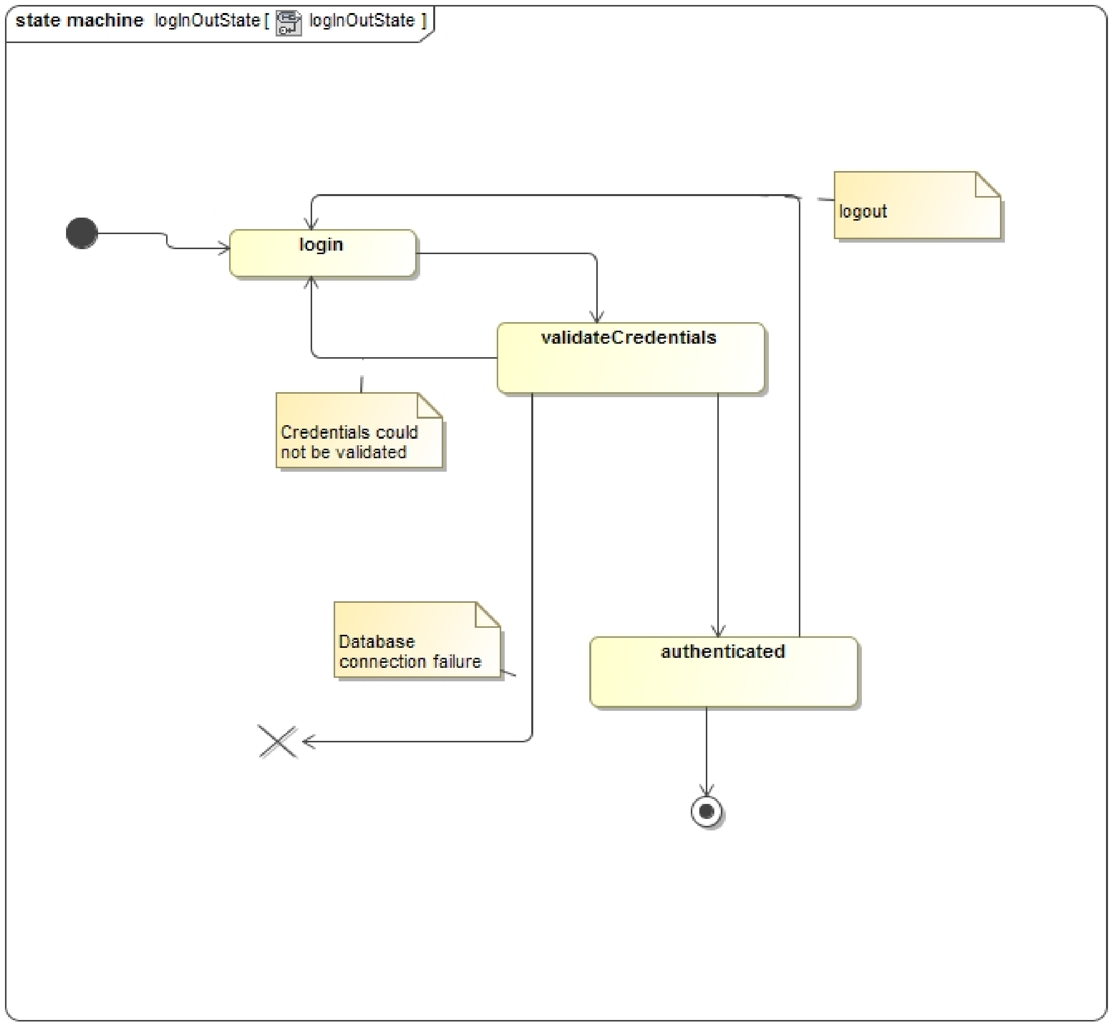
state machine logInOutState [logInOutState]
login
validateCredentials
authenticated
logout
Credentials could not be validated
Database connection failure

Figure 4: Login/logout state diagram

2.3 Project Module

This module deals with the configuration of projects. Initially a skeleton project will be set up with basic criteria and additional criteria can be added at a later stage as needed.

2.3.1 Use-cases

The project module provides services to create and manipulate projects.

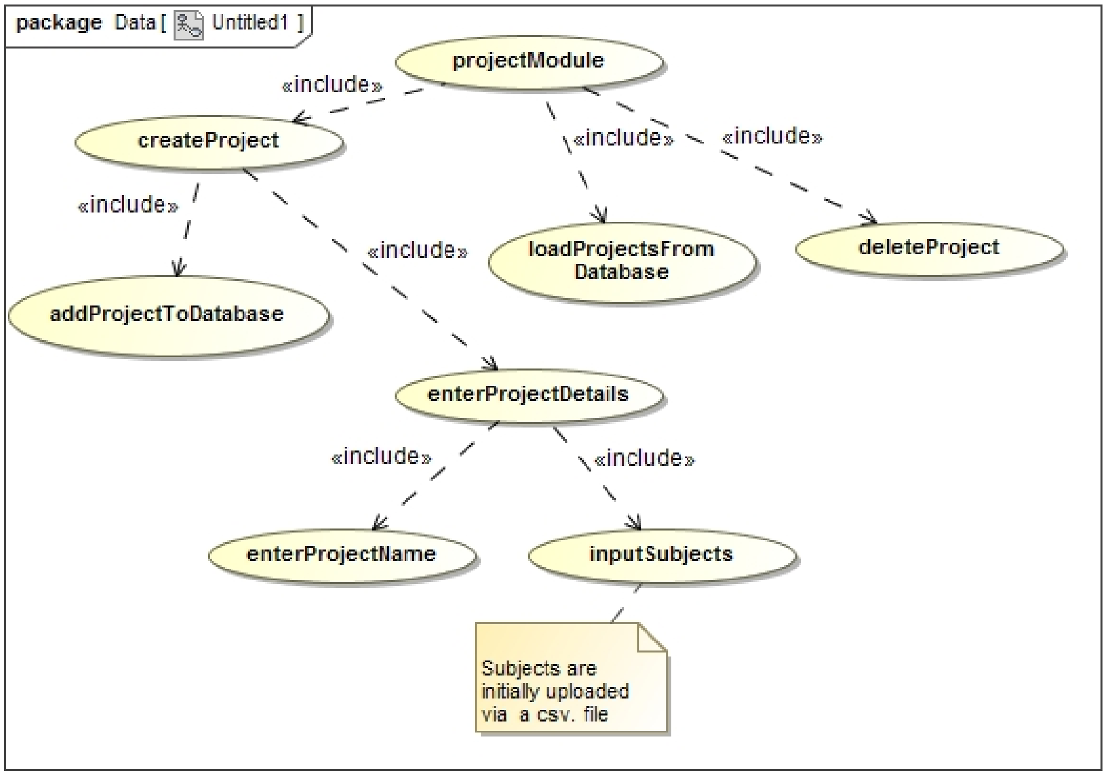

Figure 5: Project Module Use Case

1. createProject

   Priority: Critical

   Pre-condition: Client must be logged in and must have a Storm profile.

   Pre-condition: The name of the project should be unique on a user level.

   Post-condition: Project sceleton is created in the database and an initial subject list exists in the database

2. addProjectToDatabase

Priority: Critical.

Pre-condition: Client must have a STORM profile and must be logged in.

Pre-condition: A project should exist and should have subjects that was added by the .csv file.

Pre-condition: The project should have a unique name that is not the same as another project for the user.

Post-condition: Project is added to the database with the subjects and criteria.

3. inputSubjects

   Priority: Critical.

   Pre-condition: Client must have a STORM profile.

   Pre-condition: Client must have created a STORM project skeleton.

   Pre-condition: A .csv file should exist and should be selected.

   Pre-condition: The .csv should have subjects in it and adhere to upload specifications.

   Post-condition: Project database is updated with a list of subjects.

4. loadProjectsFromDatabase

   Priority: Critical.

   Pre-condition: Projects must exist in the database.

   Pre-condition: The user must be logged in.

   Post-condition: My projects page is populated with current user's projects.

5. uploadCSVToUpdateSubjects

   Priority: Critical.

   Pre-condition: Client must have a STORM profile and must be logged in.

   Pre-condition: A project should exist and should have subjects that was added by the .csv file.

Pre-condition: The .csv should be in the correct format as can be seen in the User Manual, when uploading.

Post-condition: Updated Criteria and Subjects are added to the project.

6. EditSubjectsIndividually

   Priority: Important.

   Pre-condition: Client must have a STORM profile.

   Pre-condition: Client must be logged in and authorized.

   Pre-condition: A project should be selected.

   Pre-condition: The criteria should be updated with valid values.

   Post-condition: The subjects in the project is updated individually.

7. AddSubjectsIndividually

   Priority: Important.

   Pre-condition: Client must have a STORM profile.

   Pre-condition: Client must be logged in and authorized.

   Pre-condition: A project should be selected.

   Pre-condition: The subject should be added with valid values and criteria.

   Post-condition: The subject is added to the project.

8. RemoveSubjectsIndividually

   Priority: Critical.

   Pre-condition: Client must have a STORM profile.

   Pre-condition: Client must be logged in and authorized.

   Pre-condition: A project should be selected.

   Pre-condition: The subject should be selected and deleted.

   Post-condition: The subject is removed from the project.

2.4 Database Interaction Module

This module deals with the creation and interaction with the database.

2.4.1 Use-cases

After a project has been created, a database collection is created and associated with it. The project can then request and update the database and persist as the shuffling criteria grows.

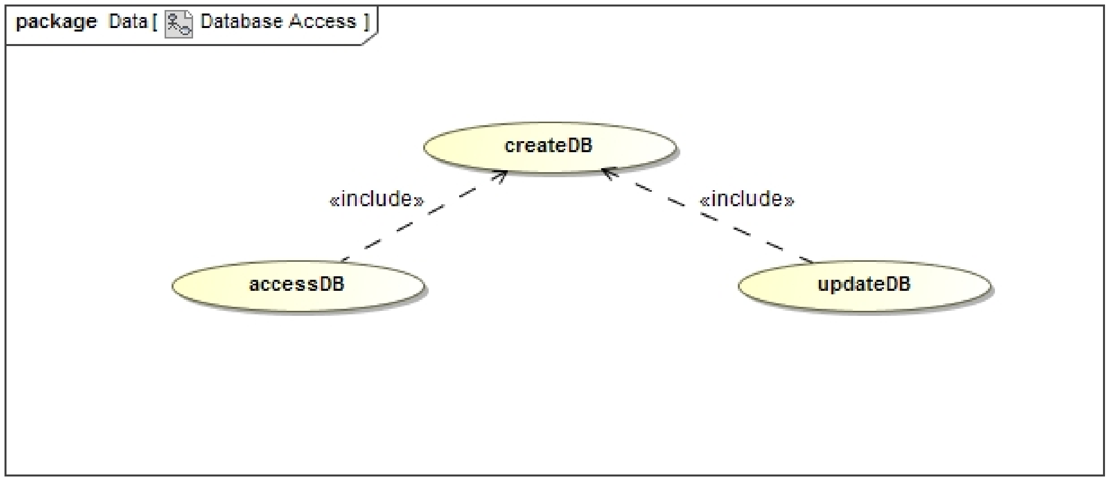

Figure 6: Database access use case

1. createDB

   Priority: Critical.

   Pre-condition: The databse should not exist yet.

   Pre-condition: Client must have a STORM profile.

   Post-condition: The database is created.

2. accessDB

   Priority: Critical.

   Pre-condition: The database should exist.

   Pre-condition: The user should have a STORM profile.

   Pre-condition: The user should be authorized to access the database.

   Post-condition: The database is accessed and queried.

3. updateDB

   Priority: Critical.

   Pre-condition: The database should exist.

   Pre-condition: The user should have a STORM profile.

   Pre-condition: The user should be authorized to update the database.

   Post-condition: The database is updated.

2.5 Algorithm Module

This module deals with the main process behind STORM, it is the shuffling algorithm.

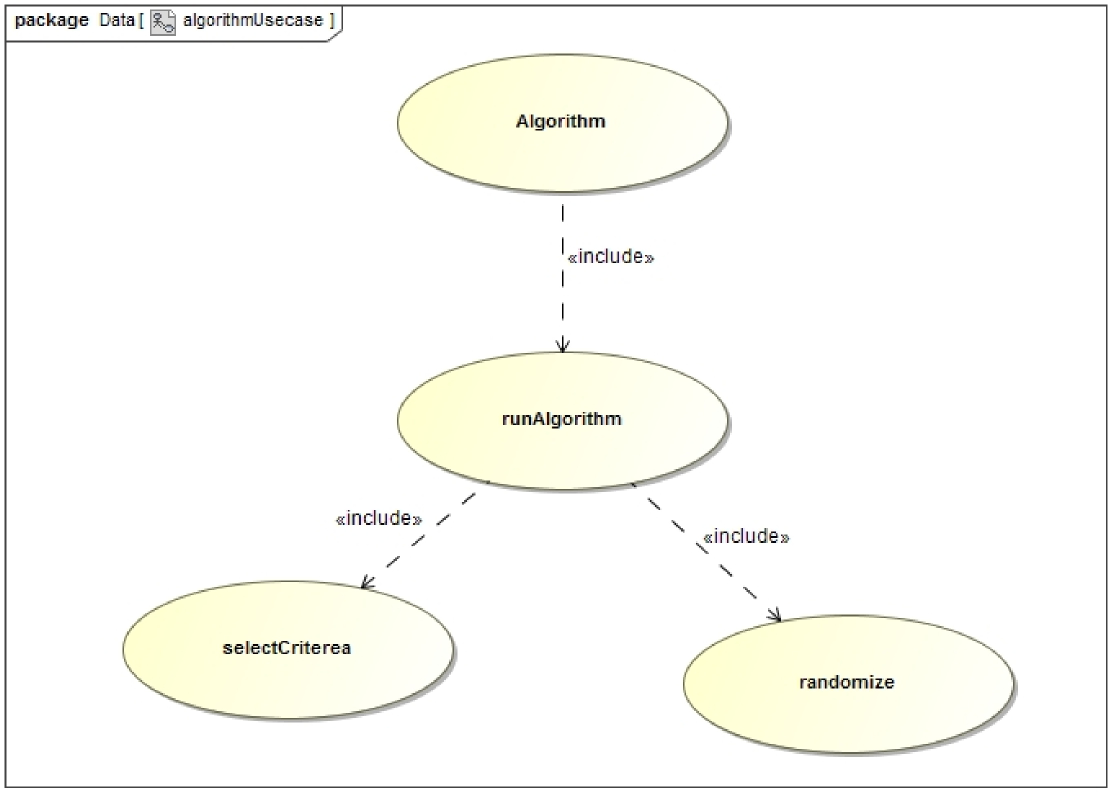

Figure 7: Algorithm module use case

2.5.1 Use-cases

The algorithm module adds the functionality required to shuffle subjects into teams.

1. RandomShuffle

   Priority: Critical.

   Pre-condition: Project must have users.

   Pre-condition: Team size or number of teams must be specified.

   Post-condition: Random teams are built.

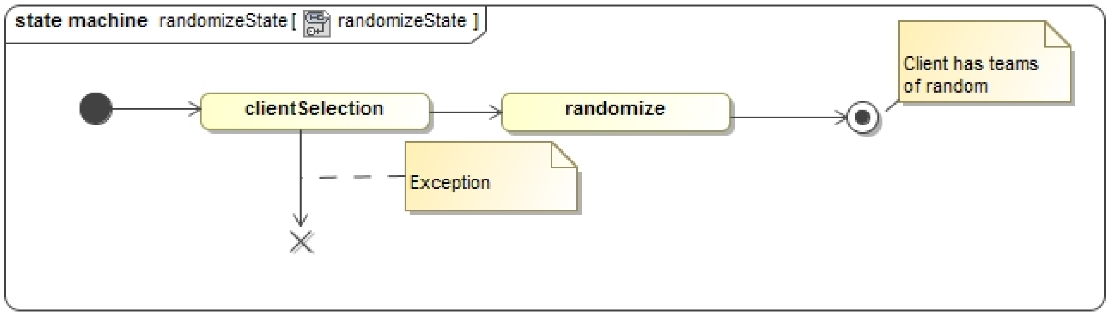
state machine randomizeState [randomizeState]
clientSelection
randomize
Client has teams of random
Exception

Figure 8: Randomize state diagram

2. SimilarShuffle

   Priority: Critical.

   Pre-condition: Project must have users.

   Pre-condition: Team size or number of teams must be specified.

   Pre-condition: Criteria with weights should be selected.

   Pre-condition: Similar shuffle should be specified.

   Post-condition: Similar teams are built.

3. DiverseShuffle

   Priority: Critical.

   Pre-condition: Project must have users.

   Pre-condition: Team size or number of teams must be specified.

Pre-condition: Criteria with weights should be selected.

Pre-condition: Diverse shuffle should be specified.

Post-condition: Diverse teams are built.

2.6 File Handling Module

This section discusses the handling, parsing and validation of imported and exported .csv files.

2.6.1 Use-cases

The file handling module provides services to import, export and validate subject set files in .CSV format.

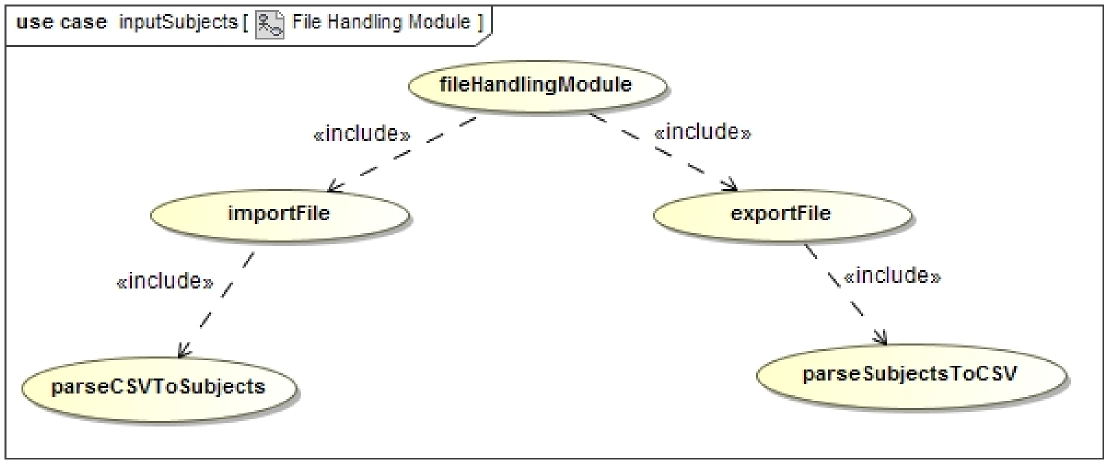

Figure 9: File Handling Module Use Case

1. parseCSVToSubjects

   Priority: Critical

   Pre-condition: The input file must pass the validation tests.

   Post-condition: Subject set is modified locally.

2. saveSubjectsToDatabase

   Priority: Critical.

   Pre-condition: The user must click on the save button.

   Pre-condition: The subject set needs to be locally modified.

   Post-condition: The subject set is saved to the database.

3. parseSubjectsToCSV

   Priority: Critical.

   Pre-condition: The subject set must exist in the database.

   Pre-condition: The correct format should be generated.

   Pre-condition: The user must have enough hard disk space to save the .CSV file.

   Post-condition: A CSV file with all the subjects existing in the project is downloaded to the user's hard drive .