MUSTE server protocol

Herbert Lange

Document version 0.0.1

1 Design principles

- stateful - authentication happens once
- unidirectional - client sends request, server answers

2 Message format

- Client messages (CM) are messages from the client sent to the server

Message Name	Valid Responses	Description
CMLoginRequest	SMLoginSuccess SMLoginFail	Send login request
CMMOTDRequest	SMMOTDResponse SMSessionInvalid	Request a Message-of-the-day, e.g. the user survey from the server
CMDataResponse	SMDataReceived SMDataInvalid SMSessionInvalid	Send result of the survey
CMLessonsRequest	SMLessonsList SMSessionInvalid	Request available lessons
CMLessonInit	SMMenuList SMLessonInvalid SMSessionInvalid	Start a new lesson
CMMenuRequest	SMMenuList SMLessonInvalid SMSessionInvalid	Send request for menus
CMLogoutRequest	SMLogoutResponse	Ends a session

- Server messages (SM) are messages from the servert sent to a client

Message Name	Description
SMLoginSuccess	Login successful
SMLoginFail	Login failed
SMMOTDResponse	A potential html-fragment for a message of the day
SMSessionInvalid	Invalid Session
SMDataReceived	Data received
SMDataInvalid	Invalid data
SMLessonsList	Lesson listing
SMLessonInvalid	Invalid lesson
SMMenuList	List of possible menus in a lesson

3 Message Datatypes

General format:

```
{"message":string, "parameters":object}
```

With `message` field containing the message name and `parameters` containing (optional) message parameters.

Message	Parameters
CMLoginRequest	`{"username":string,"password":string}`
CMMOTDRequest	`{"token":string}`
CMDataResponse	`{"token":string,"context":string, "data":["field":string,"value":string]}`
CMLessonsRequest	`{"token":string}`
CMLessonInit	`{"token":string,"lesson":string}`
CMMenuRequest	`{"token":string,"lesson":string,"clicks":number,"time":number, "a":{"tree":string,"lang":string}, "b":{"tree":string,"lang":string} }`
CMLogoutRequest	`{"token":string}`
SMLoginSuccessful	`{"token":string}`
SMLoginFail	`null`
SMMOTDResponse	`{"filename":string}`
SMSessionInvalid	`{"error":string}`
SMLessonsList	`{"lessons":[{"name":string,"description:string, "exercisecount":number,"passed":boolean}]}`
SMMenuList	`{"lesson":string,"passed":bool,"clicks":number, "a":{"lang":string,"tree":string, "lin":[{"path":[number],"lin":string,"matched":[number]}], "menu":{:[[{"score":number, "lin":[{"path":[number],"lin":string}]}]]}}, "b":{"lang":string,"tree":string, "lin":[{"path":[number],"lin":string,"matched":[number]}], "menu":{:[[{"score":number, "lin":[{"path":[number],"lin":string}]}]]}}}:`
SMLessonInvalid	`null`
SMDataReceived	`null`
SMDataInvalid	`{"error":string}`
SMLogoutResponse	`null`

1 `token` is an identifier assigned to the client session by the server. `context` defines the semantics of `data`.

- For *startQuestionaire* and *finalQuestionaire*: `field` can be one of *Field1* to *Field20* and `value` can either be a number between 1 and 5 for fields with Likert scale and a string for the freeform fields
- For *finishedSession* and *canceledSession*: `field` is *PlayTime* and `value` is the time used for completing the session or before canceling and going back

2 The token to be used by all following client requests

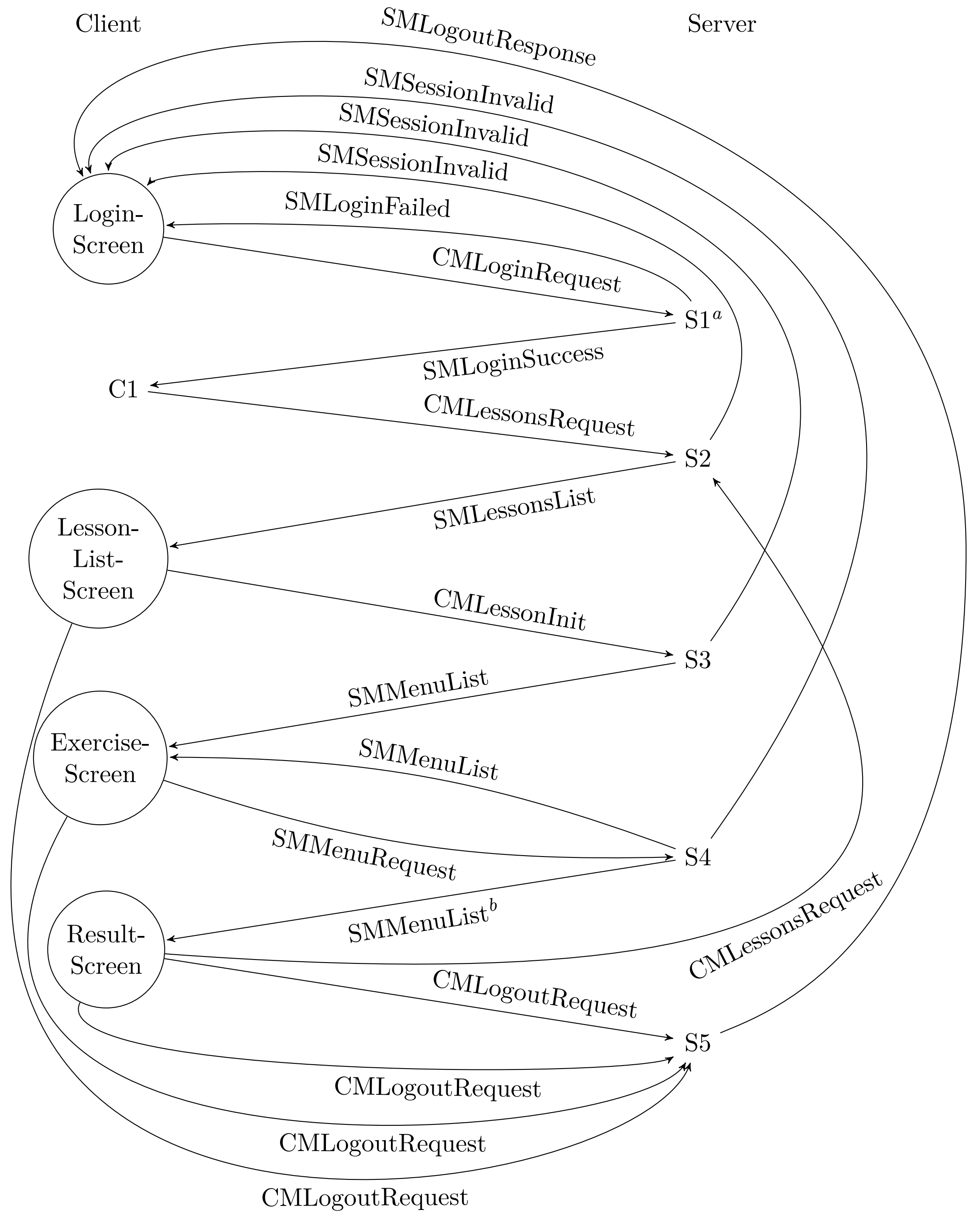
3 A file name to be displayed as a message of the day

4 Reason like timeout or not authenticated

5 lessen *name* and *lesson* are the same as the name of the PGF used for the lesson

6 Potential error message

4 Sequences

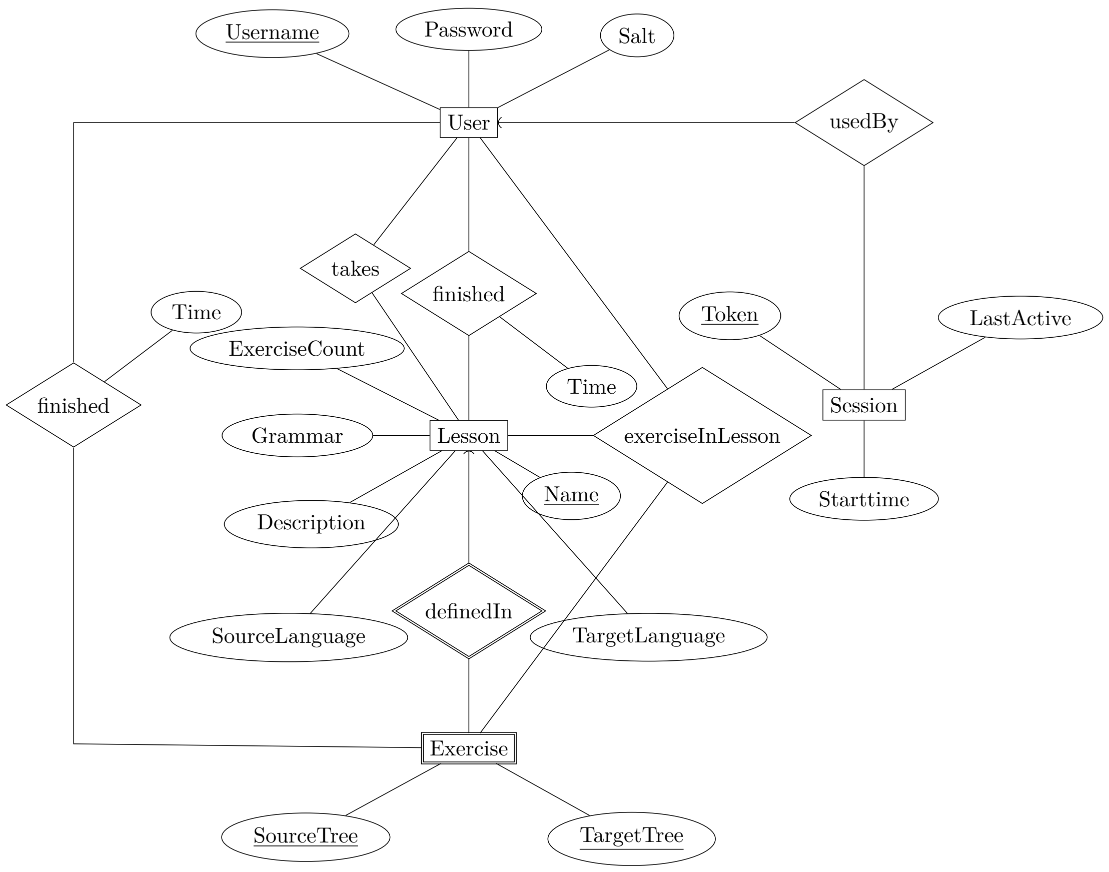


[a]S1,..,S5,C1 are hidden states

[b]If session is passed i.e. if both trees are the same

5 Database

5.1 ER-Diagram



5.2 Schema

User(<u>Username</u>,Password,Salt)

Session(<u>Token</u>,User,Starttime,LastActive)

User → User.Username

Lesson(<u>Name</u>,Description,Grammar,SourceLanguage,TargetLanguage,ExerciseCount)

Exercise(<u>SourceTree</u>,<u>TargetTree</u>,<u>Lesson</u>)

Lesson → Lesson.Name

FinishedExercise(<u>User</u>,<u>SourceTree</u>,<u>TargetTree</u>,<u>Lesson</u>,Time,ClickCount)

User → User.Username

(SourceTree,TargetTree,Lesson) → Exercise.(SourceTree,TargetTree,Lesson)

StartedLesson(Lesson,User)

User → User.Username

Lesson → Lesson.Name

FinishedLesson(Lesson,User,Time)

User → User.Username

Lesson → Lesson.Name

ExerciseList(User,SourceTree,TargetTree,Lesson)

User → User.Username

(SourceTree,TargetTree,Lesson) → Exercise.(SourceTree,TargetTree,Lesson)

Lesson → Lesson.Name

5.3 SQLite SQL

```
CREATE TABLE User (
  Username TEXT,
  Password BLOB,
  Salt BLOB,
   PRIMARY KEY(Username)
  );
CREATE TABLE Session (
  Token TEXT,
  Starttime NUMERIC DEFAULT CURRENT_TIMESTAMP,
  LastActive NUMERIC DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY(Token)
  );
CREATE TABLE Lesson (
  Name TEXT,
  Description TEXT,
  Grammar TEXT,
  SourceLanguage TEXT,
  TargetLanguage TEXT,
  ExerciseCount NUMERIC,
  PRIMARY KEY(Name)
  );
CREATE TABLE Exercise (
  SourceTree TEXT,
  TargetTree TEXT,
  Lesson TEXT,
  PRIMARY KEY(SourceTree, TargetTree, Lesson),
  FOREIGN KEY(Lesson) REFERENCES Lesson(Name)
  );
CREATE TABLE FinishedExercise(
  User TEXT,
  SourceTree TEXT,
  TargetTree TEXT,
  Lesson TEXT,
  Time NUMERIC,
  ClickCount NUMERIC,
  PRIMARY KEY(User,SourceTree,TargetTree,Lesson),
  FOREIGN KEY(User) REFERENCES User(Username),
  FOREIGN KEY(SourceTree,TargetTree,Lesson) REFERENCES Exercis(SourceTree,TargetTree,Lesson)
  );
CREATE TABLE StartedLesson (
  Lesson TEXT,
  User TEXT,
  PRIMARY KEY(Lesson,User),
  FOREIGN KEY(Lesson) REFERENCES Lesson(Name),
  FOREIGN KEY(User) REFERENCES User(Username)
  );
CREATE TABLE FinishedLesson(
  Lesson TEXT,
  User TEXT,
  Time NUMERIC,
  FOREIGN KEY (User) REFERENCES User(Username),
  FOREIGN KEY (Lesson) REFERENCES Lesson(Name)
  );
CREATE TABLE ExerciseList(
  User TEXT,
  SourceTree TEXT,
  TargetTree TEXT,
  Lesson TEXT,
  PRIMARY KEY(User,SourceTree,TargetTree,Lesson),
  FOREIGN KEY(User) REFERENCES User(Username)
  FOREIGN KEY(SourceTree,TargetTree,Lesson) REFERENCES Exercise(SourceTree,TargetTree,Lesson
  );
```