Team 28 - Deliverable 1 UML Design fname: String {optional} [For Req7D, for UI Only] lname: String {optional} username: String <<Enumeration>> <<Enumeration>> <<Enumeration>> major: MajorType MajorType SeniorityType MessageType seniority: SeniorityType For Example... email: String ComputerScience. #Note: maybe adding a history array to store quiz ids Correct Freshman Major1 Engineering, Incorrect Sophomore English, + login(username: String) Junior Math, + register(username: String, fname, Iname, major: MajorType, seniority: Senior Biology SeniorityType, email: String) {userExists(username) == false} MajorN Grad Political Science + addQuiz(uniqueName: String, description: String, words: List<Word>, incorrectDefinitions: List <String>) {incorrectDefinitions.Size == 3*words.Size && words.Size <= 10} + removeQuiz(uniqueName: String) + practiceQuiz(uniqueName: String): quiz: Quiz {username != Quiz.author when Scoreboard called add guiz using guizHistory() } + stats(): scores:List<Score> {sorted by time} + quizHistory(): List<Quiz> scores: List<Score> Score 0..* Get All Scores for this Student + orderAlphabetically(): scores: List<Score> (Interface Realization) uniqueName: String + orderByDate(): scores: List<Score> _returns_ username: String + orderByValue(): scores: List<Score> percentage: float + filterByHasTaken(): List<Score> Reg7B called by timeStamp: DateTime student object ex: student.stats().filterByHasTaken() <<interface>> **ViewableStatistics** 介 Add / Remove uses +stats(): scores: List<Score> <<Utility>> ScoreManager Get All Scores for this Quiz ----returns ----(Interface Realization) 0...* + getFirstScore(username: String): score:Score 0..* + getHighScore(username: String): score:Score Quiz uniqueName: String {quizExists(uniqueName) == false} author: username description: String wordPair: Map<String, String> {Size <= 10} incorrectDefinitionBank: List<String> {Size = wordPair.Size*3; repeatable} Req7B words:List< Word> {shuffled && unique} Req6A Request Certain Statistics numberCorrect: int {numberCorrect= 0} Word + generateQuestions(): words: List<Word> Quiz + shuffleQuestions(): word: List<Word> Score Database word: String -Display-+ displayQuestion(): word:Word Student definitions: String[4] + checkAnswer(word: Word) : response:MessageType message: MessageType + countScore():score:Float {wordPair.Size == Length} + stats(): scores: List<Score> Req7A + getTopThreePerfectScores(uniqueName: String): List<String> {filter % == 100; get first 3 elements, ordered alphabetically & return} Req7C

Student