Anatomy and Physiology Test Preparation Application

CSC 131 - Software Engineering

Professor/Sponsor: Doan Nguyen



By:

**Shah Newaz**

**Christopher Simon**

**Marco Campos**

**1. Customer Statement and Requirements**

**Problem Statement:**

For the past few years, an increasing number of students have had the goal of acquiring a position in the medical field. Requiring copious amounts of memorization of numerous medical terminologies and key words, studying for such a career can be a daunting process. Conventional methods of note taking and reading can become mundane to students who study over long periods of time. As a result, students who study in this repetitive manner will eventually retain less information over time.

**Proposed Solution:**

With the goal of improving memorization skills through interactive, enjoyable learning, our team will be developing an application to aid in student studying, particularly for those students who are perusing a medical degree. Students who utilize the app will be able to choose from various categories of which they are currently reviewing or studying for. With the plan of developing a practice test generator for each category, students will choose the amount of questions that they desire being tested on. As our primary goal is to improve retention, test questions will be asked at random, and users will be able to select questions that they no longer want to be tested on anymore. When a run of the practice test has been completed, users will have the option of retaking the same exam with the same set of questions, or only going over the questions that they had trouble with. Users will have the ability to exit out of a test at any given time.

As none of our team members are medical students, surveying, and information research using viable online resources and textbooks, will be done in order to collect the data necessary for our application. Our information will be stored and retrieved from a database, and the application will make requests to pull data from it. Students also have the ability of reporting questions that they believe to be incorrect, so that an administrator can update the information accordingly.

**Program Structure:**

Title Page: Will have the opportunity to either navigate to the instructions page, table of contents. The application can be terminated at any time.

Instructions Page: Will display the instruction on how to use the application. From there, the user can go back to the title page, or to the table of contents.

Table of Contents: Will allow the user to go back to the Title Page, Instructions Page, or to choose from a variety of subcategories.

Subcategory Pages: Users will have the opportunity to select a desired number of questions to be tested on. During the test, for each question answered correctly, the user will earn a point. For each question answered incorrectly, the application will prompt the user whether they would like to see the correct answer or not. At the end of the test, the application will display the number of questions answer correctly out of the total number of questions included in the test. From there, the user will have the option of either repeating the same test, with the option of only retaking the incorrect questions, or going back to the Subcategory Page.

**2. Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Priority** | **Description** |
| Authentication via API | 3 | Users and Administrators must be authenticated before being able to access the API that makes requests to the database on the user’s behalf. This security protocol incorporated through the API will act as an abstraction layer in order to prevent users from modifying the database directly. |
| Termination | 2 | The user should have the ability to terminate the application at any time. |
| Choose from subcategories | 1 | The user should be able to choose the topic that they desire to be tested on. Once the topic is chosen, questions generated for that test must only come from that specified topic. |
| Choose number of questions | 1 | After selecting a subcategory, the user should be able to set the number of questions to be included on the practice test. |
| Scores List | 2 | At end of each test, the application will provide the number of correct answers over the number of total selected questions for that test. |
| Random Question Generator | 1 | The application will generate a random set of questions that will eventually be retrieved from the database when a read request is made. |
| Review Previous Incorrect Questions or Whole Test | 1 | After taking a test, the user should have the option of retaking the same test, with the option of retaking only the incorrect questions again or the whole test again. |
| Read Question | 1 | After generating a list of questions to select for the test, the application will obtain the selected questions from the database through the API, so that they can be displayed to the customer one at a time. |
| Create Question | 4 | The Admin will create new questions for the database through the API. |
| Update Question | 4 | The Admin will update the database with advised questions, provided from the users. |
| Delete Question | 4 | The Admin will delete existing questions from the database through the API. |
| Request to revise Question | 4 | The user will be able to submit a request for revisal of a question if he/she believes that it yields false information. |
| Apply updates to Application | 4 | The admin will modify the application in order to fix bugs and to add features. |

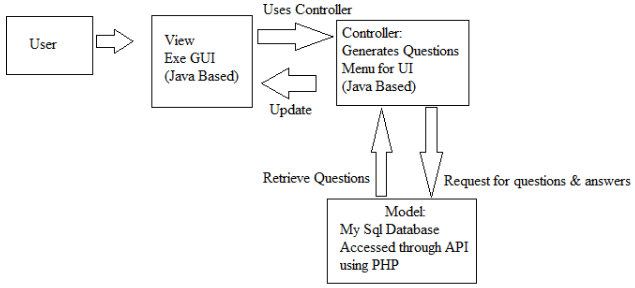
The functional requirements are split into four stages of prioritization, with stage one tasks being the most relevant to the application. Stage two tasks, such as program termination and score listing, focus on friendly user interaction, and therefore will be implemented after stage one tasks. Stage three, focusing on security, is not a task of high priority, but is better to include for a safer runtime of the application. Priority four, focusing on administrative database inquiries, may not be fully functional by the end of our initial deployment stage.

**3. Nonfunctional Requirements**

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Priority** | **Description** |
| Graphical User Interface (GUI) | 1 | The application will be visually pleasing, using textboxes, pictures, and buttons when applicable. The application must be fluid and responsive. |
| Navigation | 3 | The user will be able to navigate to whatever they need in a timely manner that does not disrupt workflow or slow down the user. |
| Interaction | 2 | Test taking will be constructed in such a way that allows for the user to review certain questions or content within a specific subcategory whenever they desire, while in the selected subcategory. |

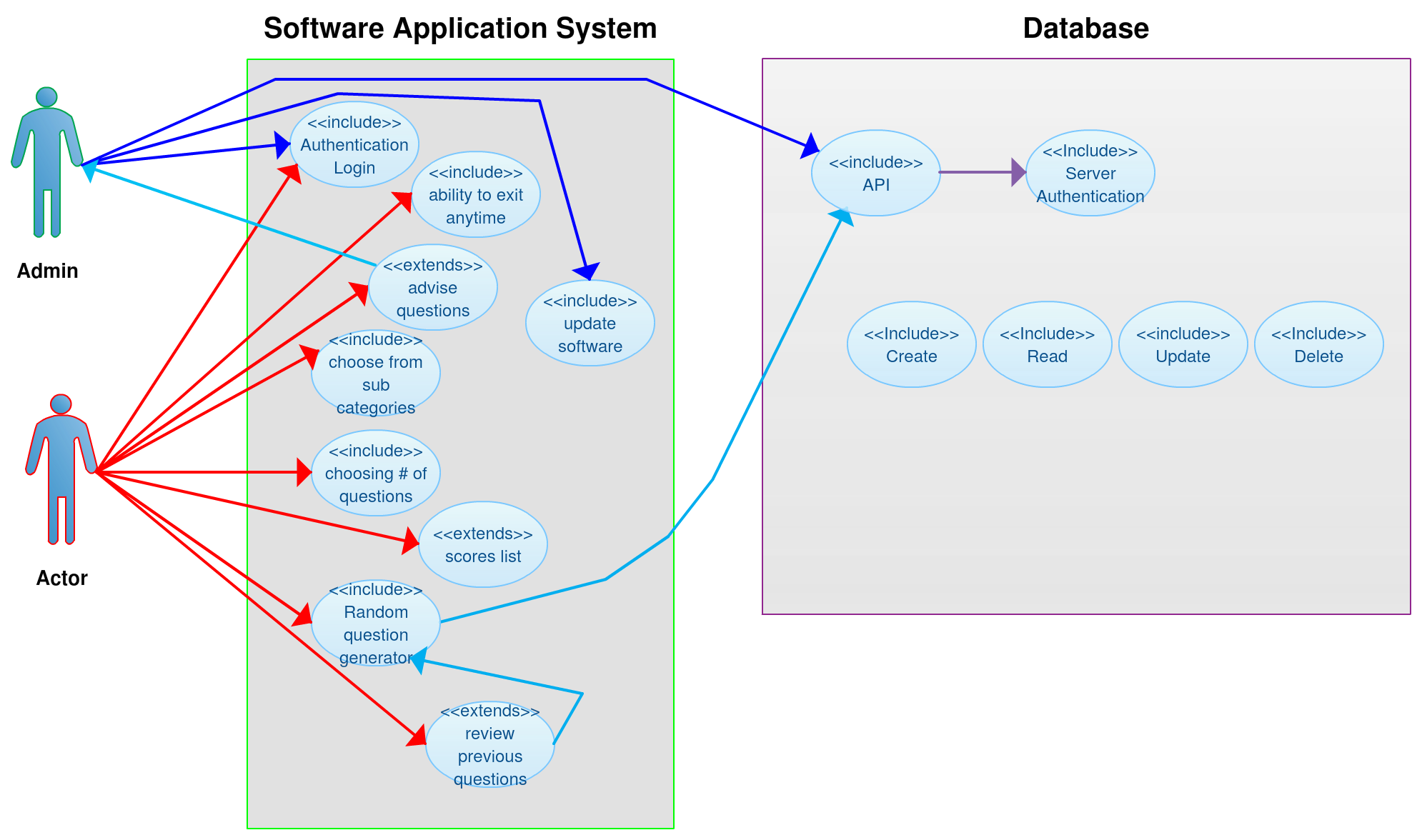
As far as nonfunctional requirements go, the GUI is most necessary as it displays the content for the user. Interaction is achieved only after the content is rendered to the user via the GUI which is why interaction is at a stage two priority. Navigation is placed as stage three, as it occurs after the interaction.

**4. System Diagram**



The system diagram represents the flow of events within the application, according to the user’s perspective. A user, when interacting with the application will first be shown the view, which contains the interface for the user to interact with. Users interact with the application using the GUI which is bound to the controller, which delegates actions between the model and the view. After the controller sends the user action to the model, the model sends the response back to the view through the controller. In this case, the model is separate from the actual application, as it is a database.

**5. Functional Requirement Specification**



The use case diagram demonstrates the whole processes of the actor and the admin interacting with the Software Application System and Database respectively. After authenticating, the admin has access to the database through the API, which allows him/her to create, update, and delete content in the database. The user, after navigating through the Application System and selecting a subcategory, will be presented with a set of questions retrieved from the database through the API. For users, extends features consist of making requests to revise questions, reviewing the previous question answered incorrectly after taking an exam, and viewing their score after an exam.

|  |  |  |
| --- | --- | --- |
| **Use Case #** | **User Case Description** | **Use Case Details** |
| 1 | User chooses a System:  a: Skeletal System  b: Muscular System  c. Integumentary System  d. Central Nervous System  e. Peripheral & Autonomic Nervous System  f. Endocrine System  g. Lymphatic and Immune System  h. Respiratory System  i. Digestive System  j. Urinary System  k. Reproductive System | Use Case Name: System  Actors: User  Priority: High  Status: Initial Development  Pre-Condition: Authenticated  Post-Condition: Current system is selected. Access to Subcategories.  Extension Points: N/A  “Used” Use Cases: Subcategory |
| 2. | User chooses a Subcategory:  a. Medical Terminology  b. Multiple Choice  c. Definitions  d. Test | Use Case Name: Subcategory  Actors: User  Priority: High  Status: Initial Development  Pre-Condition: User must have selected a system.  Post-Condition: The subcategory is selected. Selected subcategory options are displayed.  Extension Points: N/A  “Used” Use Cases: Subcategory |
| 3. | Subcategory options :  User will take a practice test, deciding how many questions they desire being tested on.  When user answers an answer correctly, they earn a point.  When user answers an answer incorrectly no point will be given and answer will be optionally displayed.  The user may quit the test at any time.  The user has the option to move to the previous menu at any time. | Use Case Name: Subcategory options  Actors: User  Priority: High  Status: Initial Development  Pre-Condition: User must have selected a subcategory  Post-Condition: User is prompted for the number of questions.  Extension Points: Taking the test.  “Used” Use Cases: N/A |
| 4 | Test:  User takes the test | Use Case Name: Taking Test  Actors: User  Priority: High  Status: Initial Development  Pre-Condition: User must have selected to take a test. User has selected the # of questions on the test.  Post-Condition: User finishes the test.  Extension Points: termination of application  “Used” Use Cases: N/A |
| 5 | Incorrect answers:  User may review all the incorrect answers from each subcategory | Use Case Name: Reviewing  Actors: User  Priority: Medium  Status: Initial Development  Pre-Condition: User must have taken a test. User must have incorrect answers.  Post-Condition: User will have gone over all the incorrect answers.  Extension Points: N/A  “Used” Use Cases: Taking the test |
| 6 | Requesting new questions: User suggests new questions to the Admin | Use Case Name: Advise questions  Actors: User  Priority: low  Status: Initial Development  Pre-Condition: User must be Authenticated  Post-Condition: Admin will have new questions to input to the database  Extension Points: N/A  “Used” Use Cases: N/A |

**6. Traceability Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Use Case Actor** | **Use Case Admin** | **Use Case Database** |
| Log in | Access to software | Access to Software and database | Authentication API |
| Navigation | Ability to move to next page | Ability to move to next page | N/A |
| Subcategories | Choosing category | N/A | The database organizes in a way that the questions will only be accessed by the category the actor chooses |
| Choose the amount of questions | Actor choose the amount of questions that will be asked | N/A | N/A |
| Correct answer display | After every wrong answer the correct answer will display | N/A | The database will include the answer only if the actor gets the answer incorrect |
| Random Question Generator | N/A | N/A | Generates the questions that will be asked |
| Score list | display the correct answers | N/A | Store the number of correct answers |
| Ability to terminate | Actor can quit at any time during test | N/A | N/A |
| Incorrect answers review | Actor will have the ability to review the incorrect answers | N/A | The database will recognize the incorrect answers and store them for later use. |
| Update questions | N/A | Admin will have the ability to update new questions and delete questions from database | The database will be updated for ambiguous questions, or needed questions |
| Read | The questions will be displayed to the actor | N/A | Generated Questions to be read by the actor |
| Create | N/A | Admin creates | The ability to add additional questions |
| Update software | N/A | Admin will update in the database | N/A |
| Advise Questions | Actor will advise questions to the admin | If advised questions are valid, questions will be added to database | N/A |

**7. Software Model**

Our project will resemble the Incremental modeling process. Initially we have discussed the effectiveness of the Agile process but since it’s our first time creating a project through repetitive documentation, we feel that the Incremental model might be more beneficial for a steady and controlled development process.

**Meetings:** We intend to meet at least twice a week to discuss how far a long we are, what questions we may have, and prepare to do in the future. A shared drive has been setup for us to keep all documented work, so it can easily be accessed.

**Time Table:**

During initial development, a time table was constructed to help the team to keep track of where we need to be. The time table is subject to change.

Week 1: Introduction

Week 2: Group Founding

Week 3: Initial Proposal Work

Week 4: Final Proposal Work/Initial SRS Work

Week 5: More SRS Work

Week 6: Completed SRS

Week 7: Research

Week 8: Reach out to potential users for additional feedback

Week 9: Database

Week 10: Programming

Week 11: Programming

Week 12: Debugging

Week 13: Improvements from debugging

Week 14: Testing

Week 15: Deploy the application to the user for use