LiveBook: Competence Assessment with Virtual-Patient Simulations

by

Sina Jalali

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Virtual-patient simulators play an important role in modern medical education. These simulators provide a safe environment for learning, give contextual feedback to learners, and allow the learner to move beyond the time and space constraints of traditional face-to-face medical instruction. In this thesis, we present an interactive simulation system, LiveBook.

This system consists of two systems. The first system generates quizzes automatically and asks the learners to answer them. They are then provided with the correct answers and their score. This helps the learners organize their basic medical knowledge.

The second system places learners in the role of a clinician who investigates the symptoms of a patient, asking questions (and receiving answers) in natural language. Once the learner has completed a case by selecting the most plausible diagnosis, LiveBook provides detailed feedback on the student's performance. The service is available at https://live-book.org. Based on our initial experience with the pilot, we believe that LiveBook can be a valuable addition to the curriculum of future diagnosticians, enabling them to apply their knowledge and develop their diagnostic problem-solving skills in the context of realistic scenarios.

Preface

All the parts of this thesis have been published in the following paper except the Speech Recognition (Section 3.7):

- Sina Jalali, Eleni Stroulia, Sarah Foster, Amit Persad, Diya Shi, and Sarah Forgie, "LiveBook: Competence Assessment with Virtual-Patient Simulations," Computer-Based Medical Systems Conference, 2017.

The software is completely developed by me and all the ideas and other contributions are the results of a collaboration of all the authors.

To my lovely wife Parisa

For her unconditional support and companion

Acknowledgements

I wish to thank Eddie Santos for his contributions to the development of an earlier prototype of LiveBook and Patrick Von Hauff for his thoughtful comments on many related discussions.

I would also like to thank Eleni Stroulia for her valuable and kind supervision. Her guidance enabled me to reach my best in this thesis.

This work has been supported by NSERC and the Faculty of Medicine at the University of Alberta.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The objective of medical-education programs is to provide future physicians with all current medical knowledge and the clinical skills that will enable them to apply this knowledge in their practice in the future. In medicine, as is the case with other professional disciplines, professional skills and competence are sometimes disconnected from classroom knowledge, and simulation has been proposed as a technique to bridge this educational gap.

According to Okuda et. al[25] Medical research and information is growing fast and medical education needs to keep up. Also patients are more concerned that students and residents are "practicing" on them, clinical medicine is becoming more focused on patient safety. Facing these challenges requires medical education to become more efficient and effective. Multiple studies have demonstrated the effectiveness of simulation in the teaching of basic science and clinical knowledge, procedural skills, teamwork, and communication as well as assessment at the undergraduate and graduate medical education levels [25]. Medical simulation provides a safe practice environment where students can practice. The systems which provide constructive feedback, also enable students to practice without the help of an instructor or an expert.

A particularly interesting simulation methodology is with standardized patients, i.e., actors playing the role of patients suffering from specific conditions, with whom medical students interact as physicians. Such simulations are motivated, in part, by the desire to expose learners to a greater variety of authentic clinical encounters than what is available to them through their typical rounds, and to help them better develop their ability to apply their knowledge, their communication skills, and their critical thinking. At the same time, such simulations are also perceived as an essential assessment tool, with the rationale that competence in realistic simulated scenarios is likely to correlate with competence in real-life practice, thus resulting in an increased patient safety.

The most important shortcoming of standardized-patient simulations is their cost: the logistical effort involved in setting them up is substantial; the financial cost of actually hiring actors to play the roles of patients can be high; an actor can only interact with a few students at a time; and, the instructor observing the students, in the case of assessment simulations, must be available for a long time for all the students to participate. The need to provide the benefits of standardized-patient simulations without their associated cost, which can potentially be prohibitive, has motivated the development of different software systems implementing virtual-patient simulations.

In our work on the LiveBook project, we have designed and developed a web-based virtual-patient simulation system that places the learner in the role of a physician encountering a patient in the emergency service of a hospital. The learner interacts with the system in natural language, asking questions and receiving the patient's answers from the system. The learner goes through the history-taking, physical-examination and lab-investigation phases, and between them, she is asked to report her most likely diagnoses and the evidence supporting them. At the end of the process, the learner is asked to decide on her final diagnosis and management plan. The system records and automatically analyzes the simulation log and provides an annotated report card to the learner as feedback and to the instructor as information useful for assessment purposes.

According to Bloom's Taxonomy [14], different educational objectives can be defined each of which requires a different level of insight into the subject. There are among the levels of Bloom's hierarchy for the cognitive domain. These levels are remembering, comprehending, applying, analyzing, evaluating, and synthesizing. Our quiz module focuses on helping the learners improve their skills in remembering and comprehending. The second system simulating the patient assists the learners mentto develop in "Applying" and "Analyzing". These are among the levels of Bloom's hierarchy for the "Cognitive" domain. The remaining levels are "Synthesizing" and "Evaluating". For a doctor to practice medicine he/she needs to have some skills in the synthesizing and evaluating levels as well. This is due to the fact that given the information about the case, they need to the synthesize that data and generate a diagnosis. They also need to be able to judge the value of the information provided to them in order to be able to process it an generate a diagnosis. LiveBook simulates the learning experience of the evaluation and synthesis levels too, but it does not provide feedback on them.

Chapter 2

Related Work

In this section we will discuss the pedagogical research. Then we will discuss the important features in the simulation software. The related software are compared to LiveBook in two tables in the next section. Then we discuss the simulation environments, case authoring environments, clinical reasoning, feedback, and natural language interaction.

2.1 Pedagogical Research

As Simulation-Based Medical Education (SBME) is getting more and more popular, numerous studies investigate the effectiveness of these systems. WC McGaghie published a critical review of SBME systems [22] and then revisited it [23] in 2016. Their conclusion they mention *"There is no longer any doubt that SBME can be a powerful educational intervention when it is used under the right conditions. The challenge for the medical education research community is to figure out how to use SBME efficiently and cost-effectively to educate and evaluate individual doctors and health care teams."* [23]. They also did a study on the comparison of SBME systems with Deliberate Practice (DP) with Traditional Clinical Education (TCE) [21]. Their study shows that SBMEs are superior to TCE in achieving specific clinical skill acquisition goals.

One simulation method for virtual patient simulation is Virtual Worlds (VW) or Second Life software. Studies show that VWs offer the potential of a new medical education pedagogy to enhance learning outcomes beyond that provided by more traditional online or face-to-face postgraduate professional development activities [30].

Simulation-based systems are costly to develop and about 1 out of 4 medical schools in U.S. and Canada produce virtual patients [12]. Hence, developing a general SBME system where developing a case is cost-effective is crucial in lowering the cost of virtual patients.

2.2 Important Features

There are four main features according to which we discuss the related software in Section 2.3 and compare them to LiveBook. These features are "what is the feedback provided to the student and how it is provided?", "does the system have branched pathways meaning do the decisions of the student affect the simulation path?", "does the system simulate the role of the doctor?", and "what is the communication method of the student and the patient?". In this section, we discuss these features further.

WC McGaghie in their work ([22] and [23]) has developed a list of important features of simulation software which help in achieving educational points most. The first and the most important factor is feedback. In the case of the systems that provide some sort of feedback to the student, it is important that the feedback is a report of the actions, potentially comparison with an expert, or it provides improvement tips for the student. This is important as the feedback, from the educational point of view, can have medical value in it.

In other papers (e.g. [2] and [26]) the importance and the potential of simulation software are highlighted in developing clinical reasoning when the system supports branched pathways. Clinical reasoning is a complex process that happens in the mind of the doctor. Having branched pathways can help developing the procedural and mechanical aspects of clinical reasoning. Yet it is an important feature of the simulation software.

Other two important features of making a simulation realistic are the simulated role and the interaction method. The first feature would be whether or not the system simulates the role of the doctor. And the second feature asks about what type of interaction the student has with the patient. The interaction could be physical, visual, or verbal communication. In the case of verbal communications, the interaction could be that the student selects the interaction from a set of options, he/she types a sentence in natural language format, or the student speaks to the patient. We have discussed the related software and compared them with LiveBook based on these four features; feedback, branched pathways, simulated role, and interaction method. This discussion is available in Appendix A.3.

We have also studied these software according to five factors; knowledge, test subjects, scope, availability, user interaction method. These factors form a high level illustrate the software. In these factors, where the knowledge comes from indicates how easily the required medical information can be acquired. The test subjects marks who is the main subject of the tool and who can be potentially educated using the system. The scope of the system demonstrates what is being simulated. Availability indicates how much effort it is required to use the system. And finally, the user interaction method is an indicator of how much it is realistic in terms of communication with the patient. These comparison are shown in Tables 2.1 and 2.2.

2.3 Related Software

Virtual-patient simulation software has been a fairly active area of research in medical education. Virtual Patients (VPs) are defined as "an interactive computer simulation of real-life clinical scenarios for the purpose of health-care and medical training, education or assessment," [9]. The trend toward the use of VPs is motivated in part by a desire to expose learners to a greater variety of clinical encounters that are available to them in real-life patient experiences and to build skills in critical thinking [5]. VPs are also being used to teach core knowledge, assess the progress of a learner, and to teach communication skills [4]. With a trend towards competency-based medical education, VPs are seen as a tool in medical education to assess learner competence and allow learners to make mistakes on virtual patients, which will potentially reduce medical errors in real-life patients [4]. While the interest in VPs is high, and their efficacy as learning tools have been established, there has been a relatively little study of the optimal virtual patient case design [7]. Evaluating a relatively optimal virtual case would provide an estimate of how simulated patients can help reduce errors in real life.

In the following tables, we have summarized comparison of LiveBook with the other software based on the important factors discussed in the previous section.

Table 2.1: Comparing related software to LiveBook

System	Knowledge	Tested with	Scope	Availability	User interaction
LiveBook	Instructor and database	Medical students	Simulates a virtual patient and the learner asks questions from the patient or orders physical and lab investigations.	Web	Natural language (free text and speaking)
Web-SP	Instructor	Bachelor students in medical, dentistry and pharmacy Courses	During each case, students can interview the virtual Patient and request lab tests to diagnose the disease.	Web	Check boxes for questions and texts for diagnosis
ISP	Instructor	MD students	Helps medical students practice in a virtual evironment With virtual patients	Windows and Mac software and web	Free-text for questions.
JDOC	Instructor	Junior medical doctors	A game designed for junior doctors to experience a busy and Hectic night in a hospital.	Application software	Mouse and keyboard. Input is in form of multiple options.
MERiTS	Instructors and database of organized knowledge	Medical students	Avatar capabilities model of the educational relevant Actions that a student can take within a virtual world.	Application software	Multiple choices or free text
faars	Not Reported	Not Reported	A game that puts students in a hospital that they have to Visit multiple patients and avoid infections and answer Questions based on the patient's disease.	Application software	Not Reported
V-PIN	Instructor	Medical students 2nd year	Simulates role of a diagnostic pathologist	Application software	Check-boxes through keyboard

Table 2.2: Comparing related software to LiveBook continued

System	**Knowledge**	**Tested with**	**Scope**	**Availability**	**User interaction**
digital clinical experience	Instructor	MSN students	It puts the student in the role of a nurse in a hospital where she interviews and examines the patient with medical devices	Application software	free text
VLE for oral health -care	Instructor + DataBase	periodontology students	learners go through the case by asking questions and providing recommendation. Their actions are saved in the system and sent to the instructor.	Web	natural language, text, media
PharmaVP	Instructor	pharmacy students	simulates different visits of a patient to the pharmacy.	Web	form
Branched-Narrative Simulator	Instructor	psychiatry students	students are given some choises through each challenge. They can see the consequences of their choices at each step.	Web	text, graphic and video
OtoTrain	Instructor	medical students	students can enter the system in two modes (study and exam), in study mode they can brows different resources and in the exam mode, there is multiple choice questions and a simulator tutorial where they can move a virtual otoscope.	Web	multiple choice question and mouse.
3D virtual dental patient	Instructor	dentistry students	The system generated 3D extraoral facial scan that is used to assist virtual diagnostic tooth arrangement process, and to gain patient's pre-treatment approval	Application software	monitor

2.4 Discussion

2.4.1 Simulation Environment

There are three types of medical experience in real-world, Face-to-Face (F2F), Question-Answer (QA) systems, and Natural-Setting systems. In F2F systems learners work with real patients. These systems use a lot of time and energy and are not sufficient in covering all the cases and scenarios that can happen. Another type includes QA systems. This category of the experiences does not improve the learner's ability in clinical reasoning and medical diagnosis. On the other hand, Natural-Setting systems are the ones that put the learner in the doctor's point-of-view and lets them decide on the questions they are going to ask, process the patient's response, and make a decision based on the gathered evidence. Another advantage of these systems is that there can be multiple cases defined in the system and the learner can go through different cases which are close cases and make the diagnosis harder. From the software discussed above all but 3D Virtual Dental Patient are QA systems. And the Natural-Setting systems include JDoc, VLE, Branched-Narrative Simulator, and ISP. We present two systems, first LiveBook quiz which is a Question-Answer system. And second, LiveBook simulation which is a Question-Answer and a Natural-Setting system.

2.4.2 Case Authoring Environment

In computer-aided education using patient simulation, the time consumed in order to make a case is an important factor in the time efficiency of the system. Therefore, user-friendly interfaces and automated processes can increase the speed of the case generation. Among the systems the ones that allow open text questions are the ones that should include more information about the patient. In order to provide a realistic environment, these systems should include responses to questions that could be asked but are not in the specific case.

The user-friendly interface helps to create cases faster as in an experiment using Web-SP [33] the authors claim that it took about 22 hours for a teacher

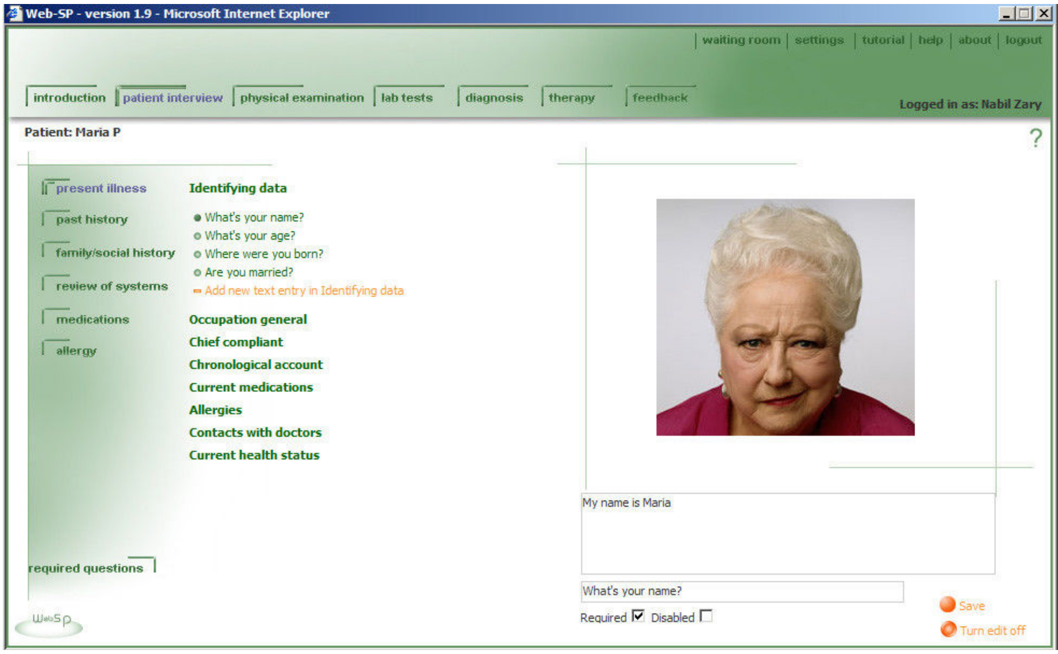


Figure 2.1: A view of the Web-SP system authoring a case

to create a dental case while a dental case takes about 2 months to be created using ISP. This is due to use of experienced multimedia producers, programmers, video editors and other people working in the team to create a case. New user-friendly interfaces which enable teachers and doctors to create cases on their own faster and easier will facilitate case generation. Web-SP, as shown in Figure 2.1, provides editable text inputs for questions and answers. LiveBook also provides editable text inputs for the answers and keywords as shown in Figure 3.7b.

2.4.3 Clinical Reasoning

Clinical reasoning is one of the most attractive features of virtual-patient medical-education systems. Forsberg et. al [10] conducted a pilot study to assess clinical reasoning in nursing using virtual patients. They used Web-SP system where similar to LiveBook, the learner goes through history taking, physical examination, lab investigations, diagnosis, and therapy suggestions. They studied on a total of 64 students and aggregated their survey assessing the system. Most of the advantages mentioned in the surveys include the help of the system in learning. The students mentioned that it was good to be able to practice on different cases and the system was realistic. They also mentioned that it forced them to think about what is important and what is not, and they could apply a great deal of their knowledge and skills. Most of the disadvantages mentioned where related to their difficulties in using the system.

One of the most important factors in medical education and clinical reasoning is the simulated role. Simulating the role of the doctor is essential as it requires the learners to make decisions without extra information or options accessible for them. In the real-world experience, the doctor does not have a set of questions he/she can ask. For example, V-PIN (Section A.3.3) and Web-SP (Section A.3.8) provide options for the learner to select from. On the other hand, ISP (Section A.3.11) and LiveBook provide open text questions. Although the possible questions are pre-defined, the learner does not have access to them.

2.4.4 Feedback

One of the main sources of learning in VP simulation software is the feedback provided by the system. Therefore, it is essential to have a comprehensive and constructive feedback. In the case of software that put the learner in the role of the doctor, the feedback could include the expert answer and feedback on learners' performance. An important part of the feedback is the part that the system or an expert evaluates the learner's performance. Most of the software that we reviewed which simulate the role of the doctor include the expert answer in their feedback. But the evaluation of the learner's performance is either delegated to the instructor or it includes a list which should have been covered by the learner. For example, in the Digital Clinical Experience (DCL) system the list of items that should have been investigated is shown to the learner at the end of the simulation as feedback. This list could also include the questions that the learner should have asked about.

LiveBook goes further than that in providing system-generated feedback. The system tracks the information provided to the learner in each asked question. For example, if the learner asks "How long did she have a fever?", the system's response would include the number of days that the patient had a fever. LiveBook keeps track of that piece of information provided to the learner. Using this information LiveBook can understand which questions can rule out potential diagnoses. Therefore, it can evaluate a question in terms of how much it would help in the process. It also requires the learner to provide the supporting evidence for a potential diagnosis. Using this information, LiveBook then understands whether or not the evidence is enough in order to support a diagnosis. This feedback along others is provided to the learner at the end of the study.

2.4.5 Natural Language Interaction

In simulating a virtual patient, an important factor is the interaction method used to interact with the patient. Some software provide options for the user when the user needs to ask a question (e.g. DCL and Web-SP). This is not the case in reality. Other software provide natural language interaction through free-text questions (e.g. ISP and VLE for oral health-care). LiveBook provides free-text questions along the capability of asking the question by speaking. This further simulates the real world where the doctor asks the question from the patient by speaking.

Chapter 3

Architecture and Process

In this chapter we are going to discuss LiveBook's software architecture and the process of generating quizzes and cases as we will explain the simulation process. In the architecture section we will explain the main components of LiveBook as we will discuss the technology used to develop the system. In the process section we will cover the generation and simulation of both quiz and simulation systems.

3.1 Software Architecture

LiveBook's overall architecture is shown in Figure 3.1. There are three main subsystems in the main LiveBook, quiz, simulation, and question interpretation.

- **Simulation System**: This system provides four services; case generation, case simulation, user-interaction recording, and report-card generation. These services are described below:
  - **Case Generation**: This service enables the website to manage case generation. It uses the Medical Knowledge Base (MKB) to enhance the experience and it creates the case in the same database.
  - **Case Simulation**: When the user starts a simulation this service is invoked. It reads the case specification from the MKB and simulates the case. It also uses question interpretation service in order to interpret the user's questions.

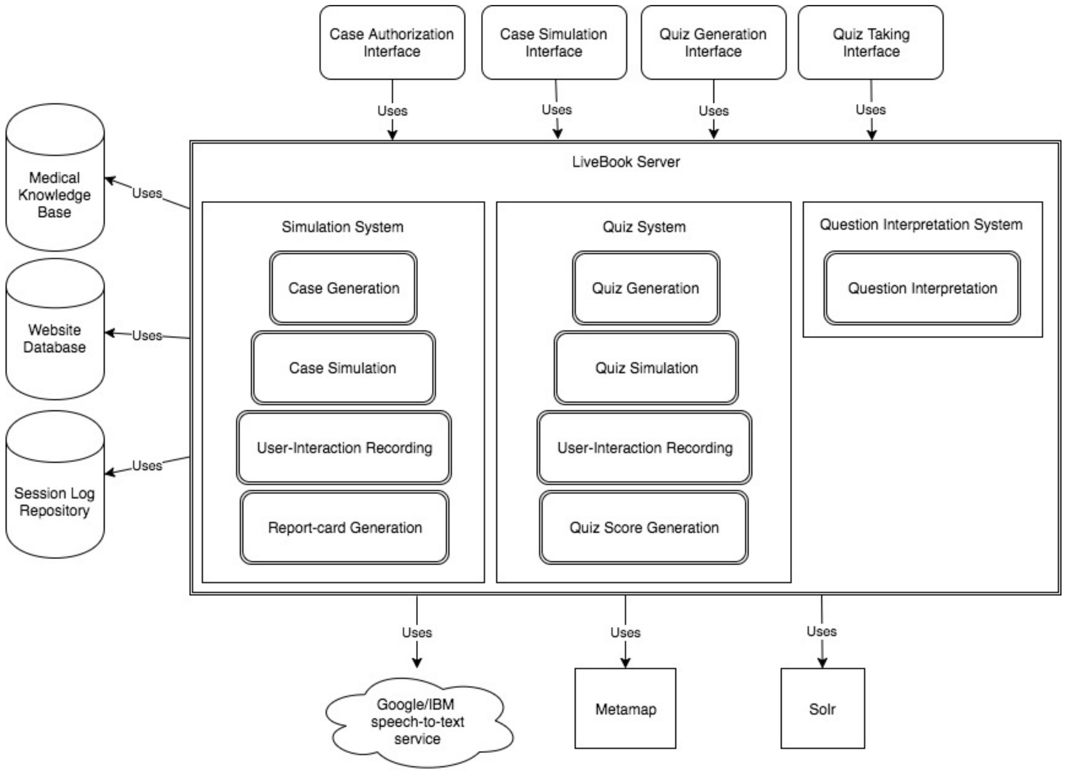


Figure 3.1: Livebook: Architecture Design

- **User-interaction Recording**: In this service, LiveBook records all the user activities when a simulation is running. It stores this data in the Session Log Repository (SLR) in order to be used later for report-card generation.

- **Report-card Generation**: After the simulation is finished, this service is invoked in order to generate a report card for the learner and the instructor. This service will retrieve the data stored in the SLR and it generates a report-card.

- **Quiz System**: This system also provides four services; quiz generation, quiz simulation, user-interaction recording, and quiz score generation. These services are described below:

  - **Quiz Generation**: The instructors generate quizzes using this service. It uses the data in MKB in order to automatically generate questions. The generated quizzes are stored in the website database.

- **Quiz Simulation**: When the user starts taking a quiz, this service retrieves the quiz from the website database and provides the required information.
- **User-interaction Recording**: When the user is taking a quiz, this system records the activities like it records the activities and stores the gathered data in the SLR.
- **Quiz Score Generation**: This service reads the completed quiz from the SLR and scores the response based on the MKB.

• **Question Interpretation System**: This system is used by the case simulation service. Its responsibility is to interpret a spoken or typed question. It matches the question with a topic in the case in the MKB. In order to interpret a spoken question, it first converts it to text using a speech-to-text service. Right now LiveBook can use either Google's cloud service or IBM's Watson service in order to convert speech to text. Then it uses Metamap and Solr in order to understand the asked question.

These services working together provide the full set of services for the website. There are four main interfaces provided by this application; case-authoring, case-simulation, quiz generation, and quiz taking interface. The website uses these interfaces. This architecture enables the system to be developed for mobile devices easily as the applications would use the same interfaces the website uses. The main server as shown in Figure 3.1, is also integrated with three databases; medical knowledge base, session log repository, and website database.

The three systems in LiveBook are developed in three different modules (applications in Django). Therefore, they work independently and they can be extended and separated easily. This architecture also encapsulates the question interpretation system. This enables the system to change its underlying technology and services without affecting other components of the system.

The system is designed and developed following an MVC architecture. This is done with Django as a platform which provides an MVC architecture. The whole system is created with a service-oriented architecture (SOA) where each request invokes a service. In Django, each request is handled by a function in Controller which will potentially interact with the Models and generates a response. In case the request is for an HTML page, the Views are rendered which are a set of HTML templates. And in the case that the request is sent to the server via an AJAX request, the information is sent to the client via JSON in order to update the View of the client. LiveBook's APIs are designed and developed in a RESTful manner.

3.1.1 Technology Analysis

There are many languages and platforms for developing a web service. Some of the popular ones are Python and Django; Ruby and Rails; Java and JSP; PHP and Symfony; C# and ASP.NET. All these technologies can create a complete website and a full set of RESTful APIs. Each has its own advantages and disadvantages. We have selected Python and Django to develop LiveBook as their combination provides valuable features to the software. Some of their advantages are as follows:

- Python is a high-level language which makes the development process faster.
- Python code is easy to read and understand as it provides high-level capabilities. This will make the code more maintainable and the convention is that python codes do not require documentation as the code itself should be easily understandable.
- Django is a complete web-development platform which handles lots of complications in developing a web service. Some of the valuable features of Django for LiveBook are:
  - Django handles user authentication, password encryption, user sessions, and cookies. It also provides libraries for third-party authentication.

  - Django's middle-wares are layers which work as a decorator pattern for the request handler functions. These middle-wares can be utilized to create a log system without increasing coupling much.
  - It prevents Cross-Site-Request-Forgery (CSRF) attacks by using a CSRF-token in the POST requests.
  - Django makes controller code redundant of the database technology as long as the database is a relational database. It provides a Python class description of the tables, then it handles the database using those classes. This prevents SQL injection attacks while it automatically optimizes database queries.
  - It also provides a built-in administration site with a full set of features for managing the data. This enables the server administrator to view and update the data in the database. It can also be used as a view for other team members with different roles as it defines roles and permissions for each user.

Some of the disadvantages of Python and Django compared to the other technologies are:

- Python is a scripting language and is a non-type-strict language. This will prevent the compiler to detect some bugs at compile time as the types of the variables are unknown at the time.

- Django is a complete web-development framework which requires some time to be set up properly for a big project. Although it provides a single setup command, it is recommended to change the project structure for easier scale in future.

- As the data in a request can be in its header or in its body, for some specific requests, it requires the data to be specified in the header in order to be recognized.

- In HTML forms with files, it requires "enctype" attribute with the value "multipart/form-data" in order to understand files sent with the form.

- For some Ajax requests, it requires the option traditional to be true in order to receive the request parameters.

Among all the other frameworks, only Ruby on Rails is comparable with Django in terms of development speed and maintainability. All the other languages are lower level languages and do not provide the features available in Python and Django as easy. There is no special reason for selecting Python over Ruby other than I was already familiar with Python.

With all the advantages and the disadvantages of Python and Django compared to the other frameworks, I decided to use them as my development technology. But still, I needed to decide between different web-development platforms for Python. One can develop web-services with Python using Django, Flask or Tornado. The main difference between Tornado and the other two is that Tornado is a small web-service platform created to have non-blocking connections. Django and Flask accept requests with a blocking socket which limits the server to keep a large number of open connections. Tornado is a good platform for the development of chat systems, notifications, or any other services where it requires to keep a connection open for a long time. Based on this a normal website can be developed with Django or Flask and the services that require live connections can be developed with a Tornado server. Comparing Django and Flask, both provide sufficient features to develop a web-service. But Flask is a micro-framework that is developed with Unix philosophy of doing one thing but doing it well. Then the community provides most of the features available in Django. A popular idiom for comparing the two is that "Pirates use Flask, The Navy uses Django".

3.1.2 Database Design

The first question in designing a database is the type of the database we are going to use. The answer to this question correlates to the use of the database. In LiveBook there are three main data required to be stored; user profile and accounts in the website database, simulation log, and medical knowledge base.

For the website, it is common to use relational databases such as PostgreSQL and MySQL. Django supports both of these databases with a few little differences which would not affect our design. Both databases are fast enough and the provided functionality covers all our requirements. Since Django developers recommended using PostgreSQL we have used it in LiveBook.

Simulation log stores simple types of data and the main requirement is speed. Therefore, relational databases with indexing feature or document-based databases like MongoDB could be used. Since PostgreSQL supports indexing and by using the same database we have better integration of data, we used PostgreSQL for this as well. Using another database would require keeping relational IDs between records of the two databases and this would add unnecessary complexity to the software.

Medical knowledge base (MKB) could be developed in another database (DB) types as well. As the purpose of this DB was to store general medical knowledge, we looked for a database designed to model general information. One of these types of databases is RDF, created to be able to store all types of information. Initially, we used RDF to store our medical knowledge in order to enable the system to incrementally add more data types. We have used the disease-symptom network from [34] and stored the values in an RDF database. The data was served using Apache Fuseki [18], which provides a web service for accessing and editing RDF databases.

After the development of our MKB, we discovered that RDF databases were too slow as soon as the database gets bigger and we could not improve the speed by optimizing our queries further. On the other hand, relational databases were not bad at modelling complex information if the data was structured properly. Since our online search for a well-structured database of medical knowledge including the disease-symptom network failed, we decided to use our PostgreSQL for the purpose of storing our MKB too. This increased the speed dramatically. For example, a query which would take more than a second took a small fraction of a second.

3.2 Medical Knowledge Base

LiveBook performs based on a Medical Knowledge Base (MKB) which is a database of medical information. A simulation software can simulate a patient and enable the learners to ask the questions they want. But in order to be able to provide constructive feedback, the software should also understand the information provided to the learners. Based on that information, LiveBook then can evaluate their performance and provide professional feedback. Now the question is ”What type of medical information should be embedded in LiveBook?”.

Since the purpose of this information is to enable the system to evaluate a diagnosis process, the most important information is the network of diseases and their symptoms. This information is the main medical knowledge required for making a diagnosis. Clearly, this is not the only information that a doctor uses in order to make a diagnosis. An epidemic disease, statistical information about the population, and not simulated information (e.g. sense of touch in the physical examination) are extra information affecting the diagnosis but is not in the MKB. But the disease-symptom network is the main medical information a doctor needs to make a diagnosis.

The network in LiveBook is a bipartite graph with diseases on one side and symptoms on the other side (Figure 3.2). A disease is a potential diagnosis of an unhealthy patient (e.g. Otitis Media). And a symptom is a pair of property and value. For example, ”Fever duration” can be a property and ”3 days” can be a value. This property and value can be paired and form a symptom. In this example ”Otitis Media” in the graph is connected to the symptom vertex formed by the pair (”Fever duration”, ”3 days”).

Any sign, evidence, indication, or generally the manifestation of a disease or disorder can be represented in our notion. This notion can also differentiate between various diagnoses of the same disease. For example, ”Mild Acute Otitis Media” is a potentially different diagnosis than ”Mild Acute Otitis Media in a child with an anatomic malformation of the head and neck” or ”Mild Acute Otitis Media in a child of aboriginal descent”. These three diagnoses

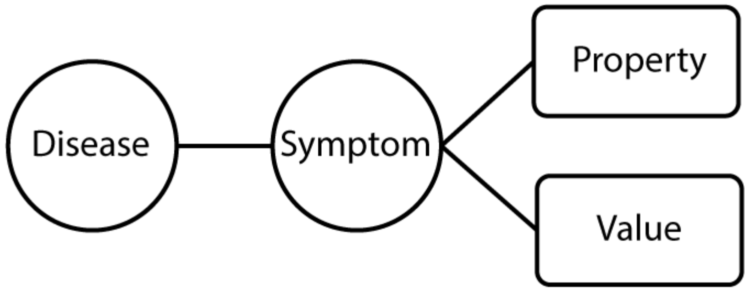


Figure 3.2: Livebook: Knowledge Network Schema

are considered different and each has its particular treatment plan. Although our notion differentiates between these diagnoses, it fails to differentiate them structurally as they are all represented with a simple string.

Currently, the knowledge base is gathered from the cases created within LiveBook which are covered in chapter 3.4. Each case in LiveBook simulates a patient with a specific diagnosis. After a case is created in LiveBook, its information is aggregated with the main knowledge base. This gradually improves the system.

3.3 Quiz Generation

As mentioned in the introduction LiveBook's quiz module focuses on the lowest levels of education according to Bloom's Taxonomy, "Remembering" and "Comprehending". Pediatricians need solid knowledge of Disease-Symptom relationships. In order to help pediatricians with this important task, LiveBook provides the quiz module. The system generates quizzes for a group of diseases automatically based on its knowledge base.

3.3.1 Quiz Generation Process

LiveBook provides the capability for the instructors and learners to generate and study quizzes respectively.

LiveBook makes it as easy as possible in order to generate a quiz. The author only needs to select a name for the quiz and a disease for which the quiz is going to be generated. The system provides the diseases already defined in the knowledge base in form of suggestions.

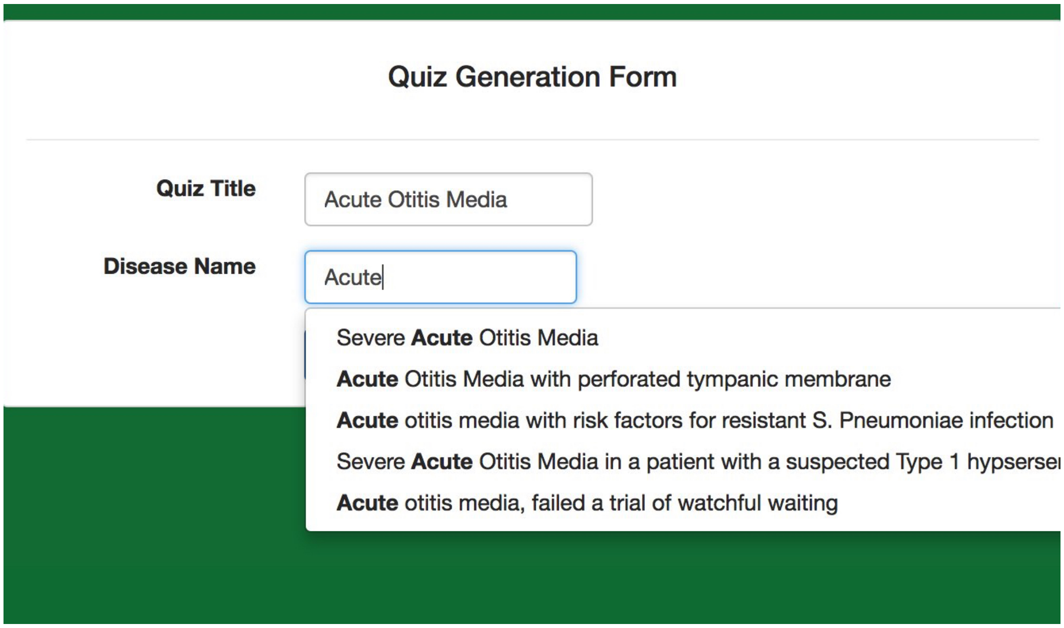
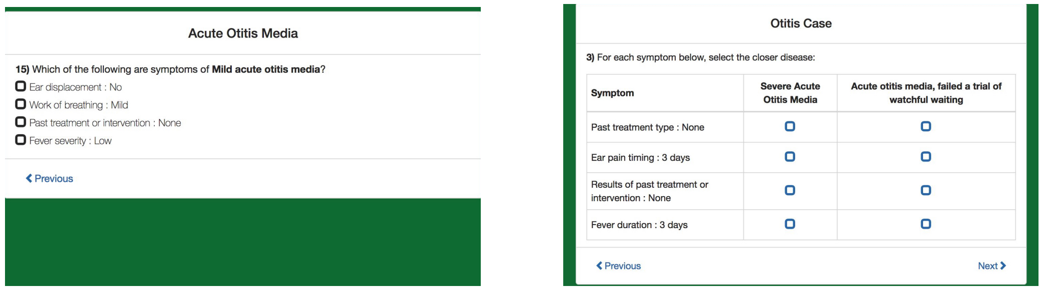


Figure 3.3: Livebook: Generating a quiz

The system generates two main types of questions. First, traditional quiz questions shown in Figure 3.4a (single-choice questions, multiple-choice questions, and true/false questions) . Second, comparing questions, the learner is given a set of symptoms and two diseases to select associated disease for each symptom (example in Figure 3.4b).

Given disease D, LiveBook needs to find a set of alternative diseases in order to generate questions. The first step is to find the set of properties the value of which is known for D. This is the same as all the properties of the symptoms associated with D. Then the system measures the distance of D with all other diseases by the Jaccard distance [15] of their symptoms. Next, it sorts the alternative diseases according to their distance and selects up to 5 closest diseases.

Having a set of alternative diseases A, generating traditional questions is simple. Here we describe the process for multiple-choice questions and the rest are generated similarly. Generating a question for a disease d in $A \cup D$, the system selects four properties associated with d. Then for each property in the set, it selects a random value. These property-value pairs or symptoms will be the options for the question.



(a) Sample multiple-choice question (b) Sample assignment question

Figure 3.4: Livebook: Two screen-shots from quiz questions

Each assignment-question is generated based on two diseases in $A \cup D$. LiveBook selects a set of mutual properties of the two diseases with different values for each disease, then each row is the symptom with the value of one of the two diseases.

3.4 Case Generation

The second system in LiveBook simulates a patient with a disease. There are three main components to this system; case authoring environment, case simulation, and simulation log analysis. In the first component, we allow the instructors create and edit cases. In the second component, the learner can study a case by asking questions, do physical examinations, and order lab investigations. At the end of the simulation, the learner is asked to provide a management strategy (e.g. discharge home, admitted to hospital, or observer in the emergency department). And in the last component where LiveBook analyses the simulation log, the learner is provided with a detailed report and a comprehensive feedback on his/her performance.

3.4.1 Patient Specification

Before creating a case we need to define our notion of a patient. In order to define a patient in our system, we only need to model health-related aspects of a person. But this model can get complex as the modelled information can potentially include past medical records, social interactions, attitude factors,

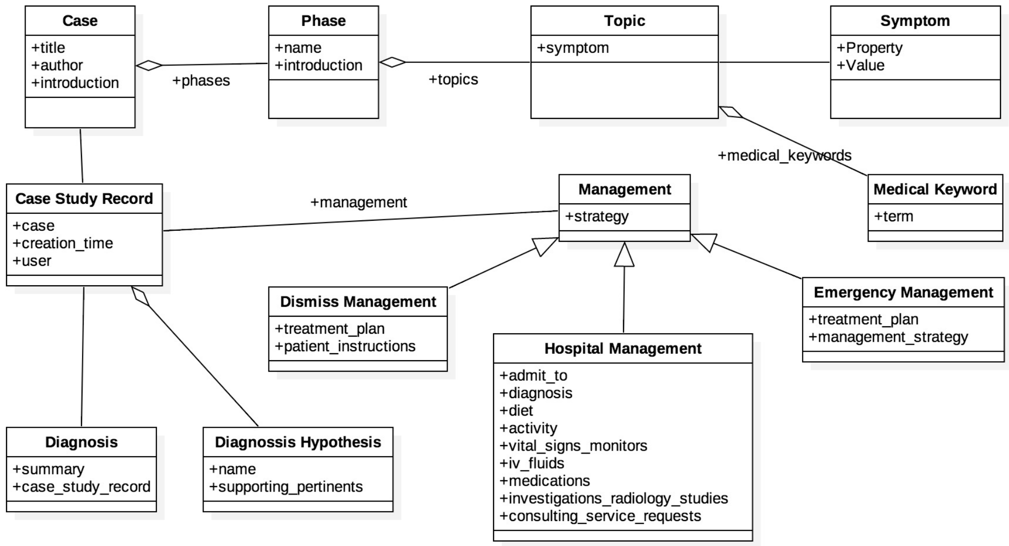
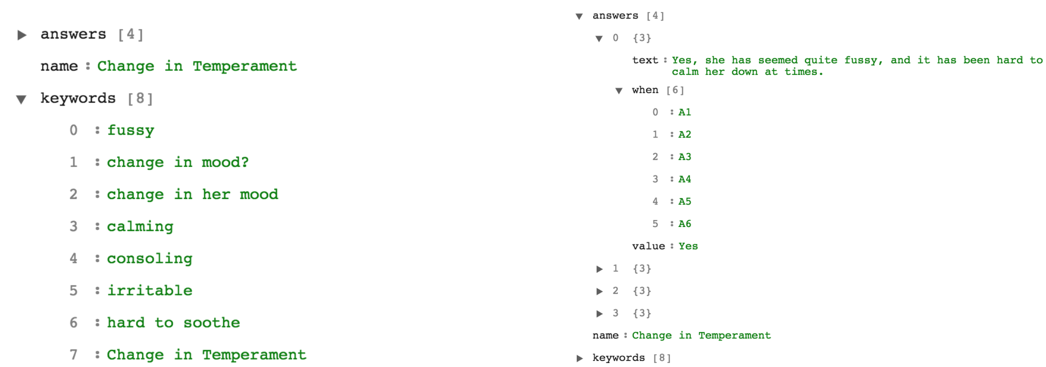


Figure 3.5: LiveBook: Case Specification UML

and physical attributes. Due to the complex nature of our problem, we took a simple and general approach which is that we represent a patient with a set of properties each with a corresponding value. This enables us to include all different types of information in our model. For example, the information "She has not been hospitalized" can be represented as "Past hospitalization records" as property and "None" as value. Or the sentence "She has lost her appetite in the past week" can be modelled using two (property, value) pairs ("Loss of appetite", "Yes") and ("Loss of appetite duration", "7 days").

3.4.2 Case Specification

The concepts of the LiveBook's case-specification language are illustrated in Figure 3.5. We have developed this language in order to provide a computer-understandable representation of a case where the author can generate a case with as much flexibility as possible. The key information in a case is a collection of relevant *topics*, related to the patient's medical history, physical status, and laboratory investigations, and their *symptom*. For example, in a particular scenario, it may be relevant to know that the patient's *body temperature* is *38 degrees* for the *last three days (fever duration)*, the *color of his skin* appears *normal*, and lab tests have revealed the presence of *Elevated White Blood Cell*



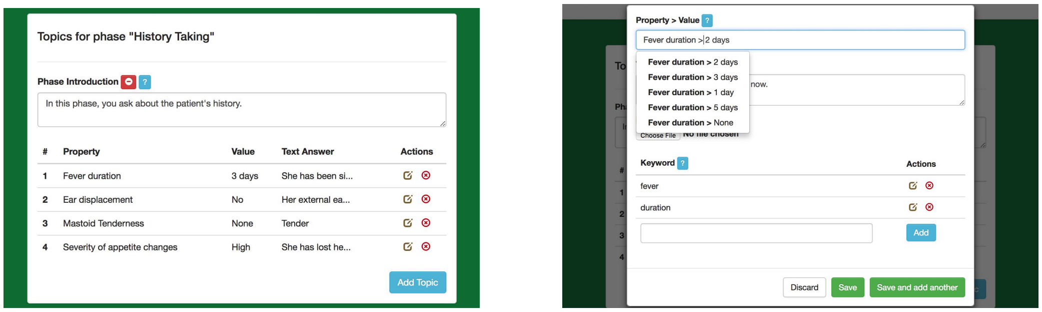
(a) Medical keywords for interpretation of the topic "Change in Temperament". Keywords common in different ways of asking about this topic

(b) Different answers for different pathways of "Change in Temperament". For pathways A1 to A6 the value is "Yes" and the text response would be "Yes, she has seemed quite fussy, and it has been hard to calm her down at times."

Figure 3.6: Livebook: A single topic in case specification with different attributes expanded

Count (WBC) in Cerebrospinal Fluid (CSF) for Cell Count.

LiveBook provides two methods to author a case. In the first method, the instructor can create a JSON file which has the content of one or more cases. Each topic in the JSON file is defined with a single JSON object shown in Figure 3.6. Using a text-based method like JSON is hard to develop and edit a case as the format can be easily broken. On the other hand, it has some advantages. For example, the author can create multiple cases (a.k.a. pathways) using a single JSON file and define one answer for multiple pathways easily. As shown in Figure 3.6b, each topic has an attribute, "answers", which lists different potential answers - A single JSON file can define multiple cases with potentially different answers to the same question. For each question, each item in "answers" has a list of pathway names for which this answer should be considered. This way an already defined topic with a specific answer can be assigned to multiple pathways just by adding their names to this list. LiveBook provides a command-line API for editing the cases in LiveBook. This API is accessible to the server administrator which requires the author to provide the case to the admin in order to be integrated with the system. The admin can create a case easily by typing the command below in a terminal



(a) Case Authoring: Editing phases (b) Case Authoring: Editing a topic

Figure 3.7: LiveBook: Case authoring

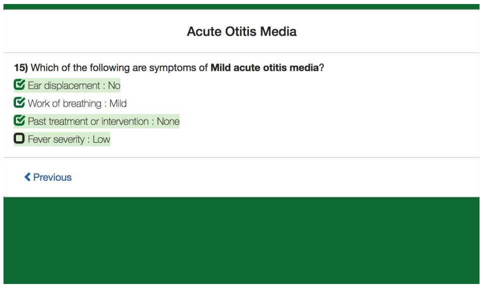
provided $PATH$ is the path to the JSON file:

```
livebook.py add_case PATH
```

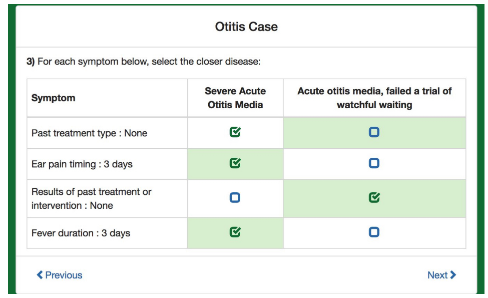
In addition to the JSON method, LiveBook provides a visual environment for creating cases. In order to create a case, the author must fill out a form including the case name, right diagnosis, and instructions. Then the author is going to define each phase. Each phase requires an introduction and a set of topics that could be investigated. Screen-shots of this process are shown in Figure 3.7. In this method, the author can easily edit the topics without worrying about breaking the JSON format (shown in Figure 3.7b). Also, this method enables the author to combine text response with an image or video. Moreover, using this method, the medical keywords for a topic is set automatically if the property has been defined already. This is done due to the fact that medical keywords related to a property is only related to the property itself and is redundant of the case. This will reduce the effort required to create a new case.

3.5 Quiz Answering

After the quiz is generated by an instructor, the learner can select the desired quiz and take it. After the learner has answered all questions, the score with the correct answers is shown to the learner. This is created so that the learner can review his/her answers and compare them with the correct (highlighted) answers. Sample screen-shots of this are shown in Figure 3.8.



(a) Sample multiple-choice question with correct answer



(b) Sample assignment question with correct answer

Figure 3.8: Livebook: Two screen-shots from quiz questions with highlighted true answers

3.6 Case Simulation

3.6.1 Simulation Steps

The process of simulating a patient starts with training the learner how to use the system. We have created a fake case and took screen-shots of the process. This will be shown to the learner before entering the simulation in order to minimize potential confusions (Figure 3.9).

In the LiveBook simulation process, the learner is then presented with the basic facts about the virtual patient, such as her name, age, gender, and presenting complaint.

Through the user interface, the learner is asked to assume the role of the physician treating the patient, and go through the history taking, physical examination, and lab investigation phases. In each phase, the learner can ask questions from a number of categories. For example, during the physical-examination phase, the learner should be asking questions and collecting information about vital signs, head and neck, and respiratory status. Each one of these categories corresponds to a set of topics of the virtual patient that the learner should be asking about. All pediatricians go through these phases and they can go back a phase in LiveBook as they could do in reality. In each phase, the learner can ask questions (Figure 3.11). After each phase, the learner is asked to provide the current differential diagnosis which asks for a finding summary and up to three potential diagnoses (Figure 3.12). These

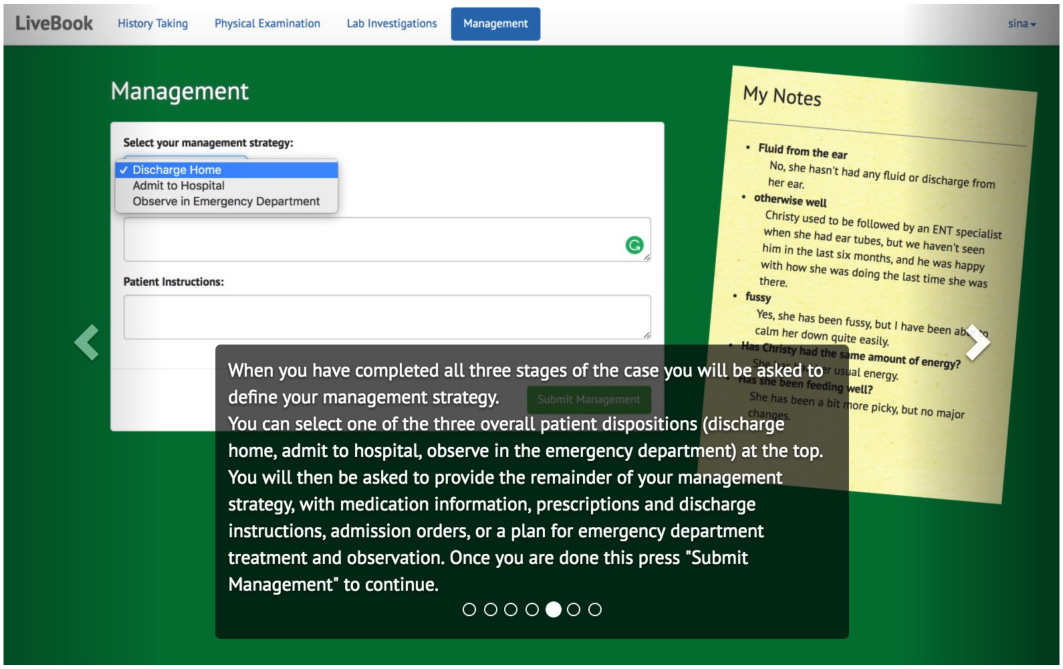


Figure 3.9: Simulation tutorial Screen

diagnoses are updated after each phase until the learner ends up with the most likely diagnosis after Lab Investigations.

After the investigation is complete the learner provides a management plan and he/she is provided with the expert's answer.

3.6.2 User Interface

The case-simulation interface is implemented as a responsive web-application accessible from any device. It consists of four main screens. The first one presents the virtual patient to the learner. The first screen, shown in Figure 3.11, indicates the current phase (at the top) and provides a text-box through which the learner asks her questions in natural language either typing or speaking to the microphone. The system provides its answer (also in natural language, as it would have been provided by the patient) just below the question text-box. If the answer includes media it will be shown below the text response. In the rare case when the system cannot properly understand the learner's question, the learner can provide the feedback that LiveBook did not interpret his/her question properly.

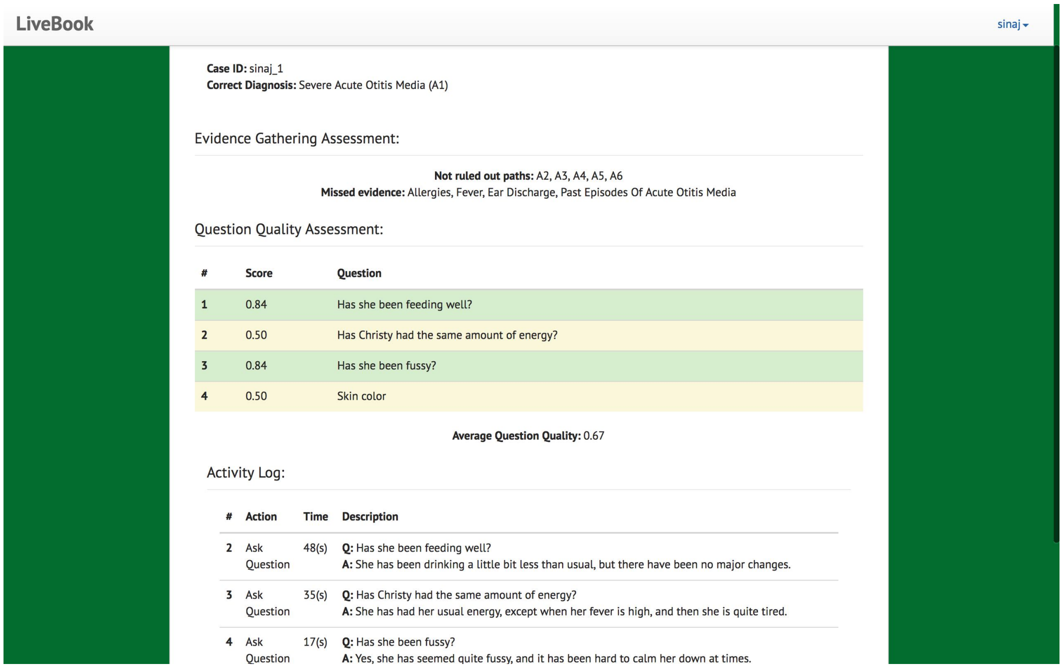


Figure 3.10: Automated Report Screen

To the right, the screen includes a notepad with all the questions already asked and their corresponding answers. This keeps the history of the asked questions and enables the learner to use this service in any place even without access to a notepad.

The third screen invites the learner to provide her top three diagnoses, and list the supporting evidence based on the values of the properties established to date, and to decide on a management plan. This screen is shown at the request of the learner, or at the end of each phase.

Finally, after the learner has gone through all phases and chosen a final diagnosis and his/her management plan. The management plan consists of a patient disposition (discharge home, observe in the emergency department or admit to the hospital), the provision of patient instructions (when to return for reassessment, what signs to consider worrisome in the coming days, etc), and medication instructions (either over the counter or prescription). If the patient is admitted to the hospital, the learner is prompted for admission orders. If the patient is observed in the emergency department, the learner is prompted to provide a rationale and plan for eventual discharge or admission.

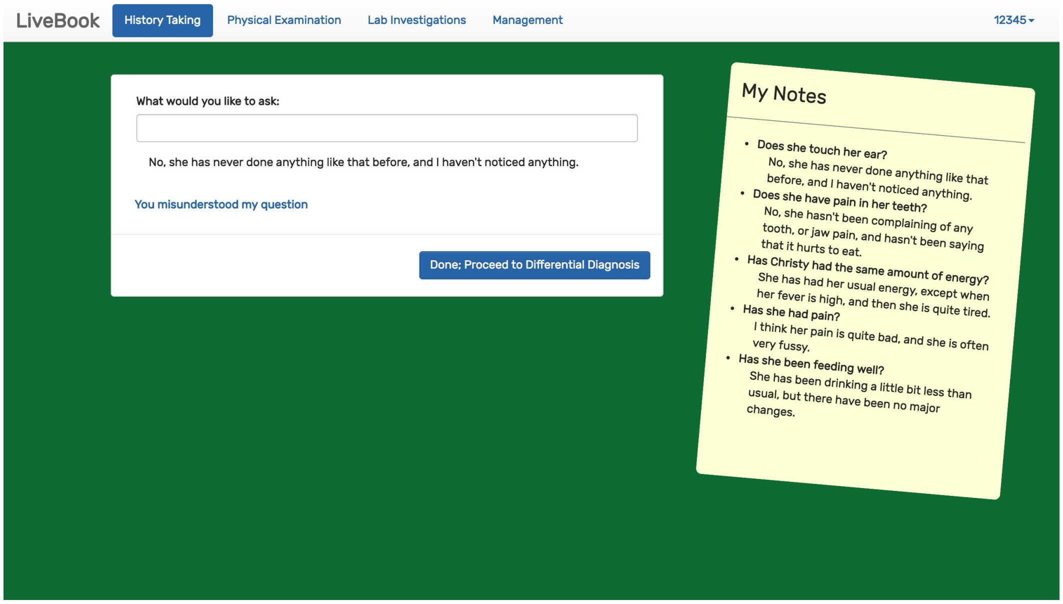


Figure 3.11: Case Investigation Screen

After a case is studied, the instructor can view the automatically generated report of a study 3.10. At the top, it shows the learners ID and correct diagnosis. Next, it shows the evidence gathering assessment. In this section, the instructor views a list of pathways (diagnoses) if the learner has not asked enough questions. The list includes the pathways that could potentially be diagnosed based on the asked questions. It will also show which topics could be investigated in order to rule out those diagnoses.

Below that, LiveBook provides quality scores for each asked question. A score is a number between 0 and 1; zero representing irrelevant questions that could not possibly provide helpful information and one representing the best possible question. And at the bottom, it shows the activity log of the learner, listing learner's actions. These actions include starting a case, asking a question, creating/updating differential diagnosis, or submitting a management strategy.

3.6.3 Recording User Activity

During simulation, the system records all the interactions between the learner and the system, and their time-stamps. This log is subsequently used to

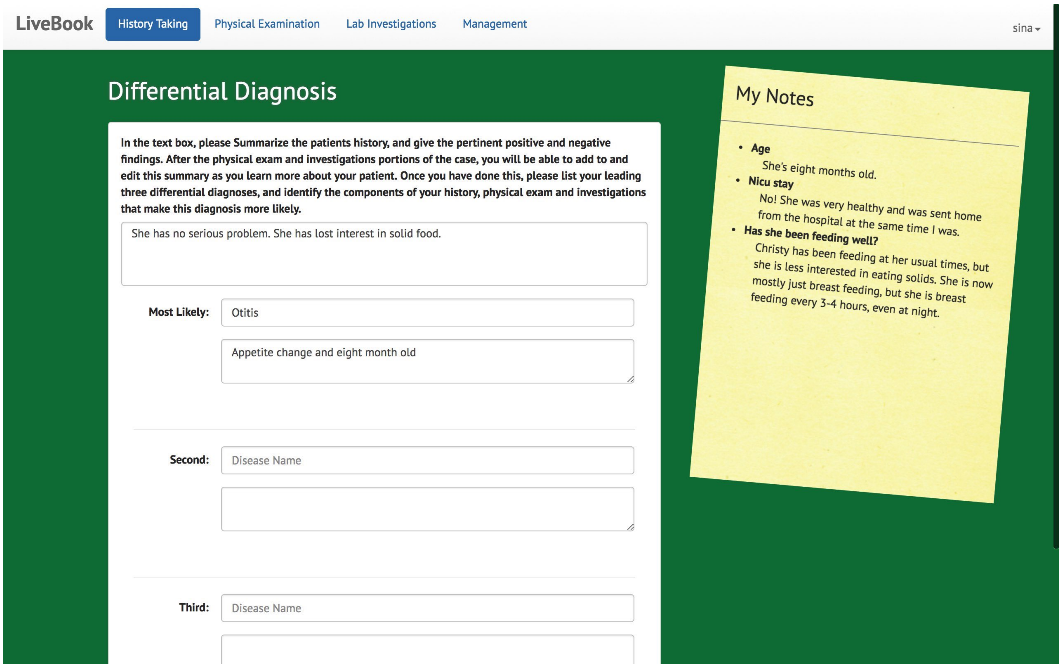


Figure 3.12: Differential Diagnosis Screen

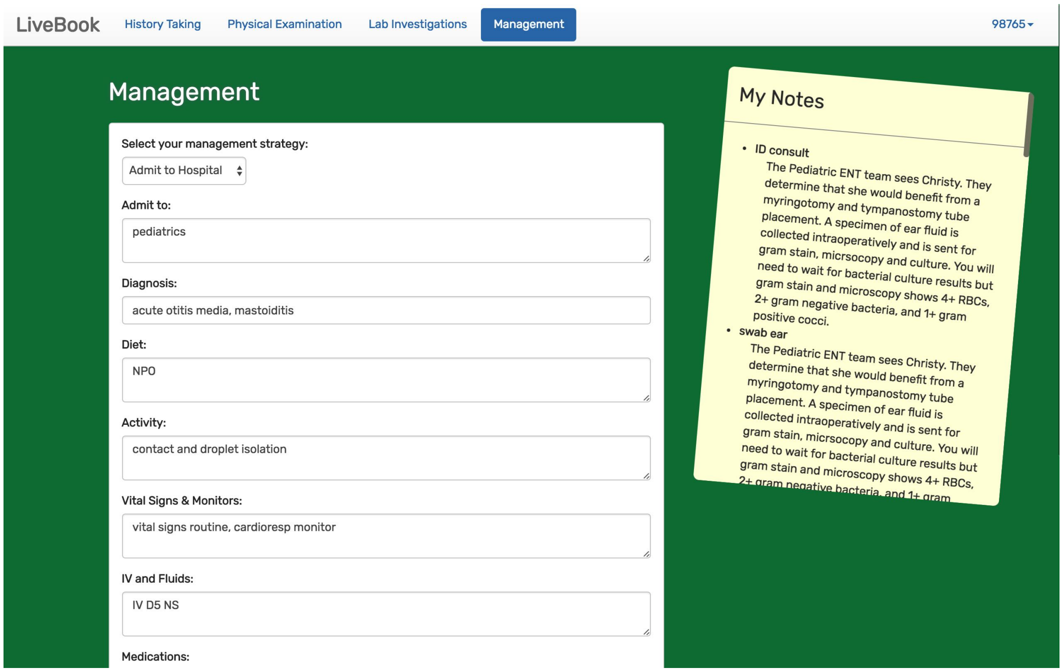


Figure 3.13: Management Screen

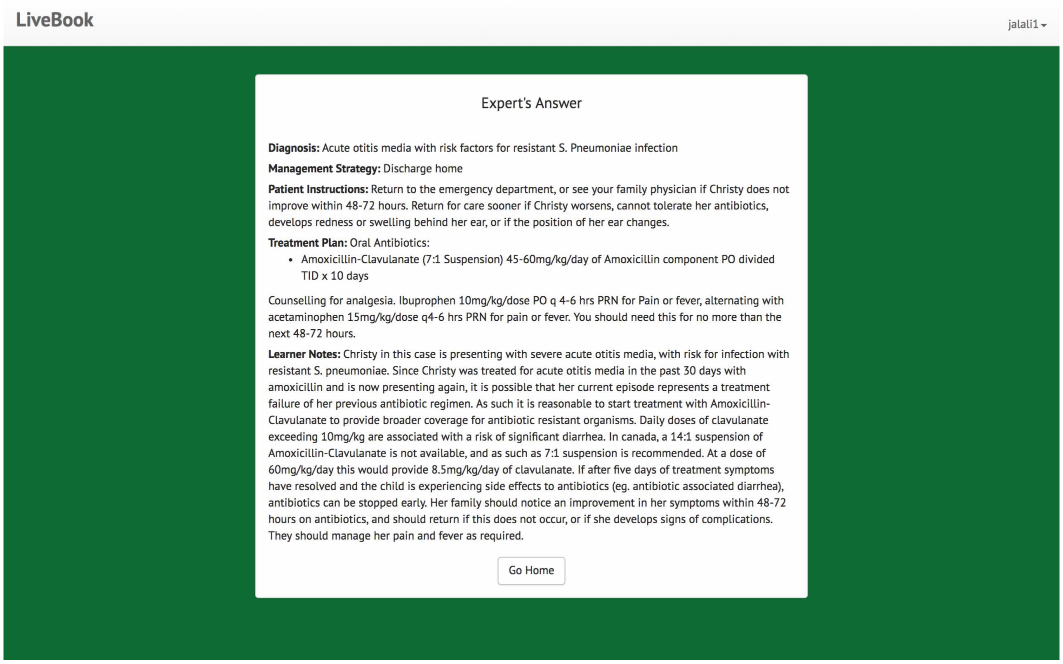


Figure 3.14: Expert Answer Screen

analyze the learner's performance and to generate reports, as shown in Figure 3.10. The automatically generated report is provided to the instructor, who can use it to inform her assessment of the learner's knowledge of facts and procedures and diagnostic-reasoning skills.

The interactions that are stored in the log are when the learner starts studying a new case, asks a question, adds a new differential diagnosis, updates a differential diagnosis, and submits a management plan. This information with the time of each action is enough to analyze and generate a comprehensive feedback on the learner's performance.

3.6.4 Question Interpretation

A key decision in the design of LiveBook, and an important contribution of our work, is its naturalistic interaction model: the learner is placed in the role of the physician and she performs that role by asking the virtual patient questions in English and interpreting the patient's answers, also in English (much as she would in the real world).

This interaction is made possible through a collection of pre-stored keywords for each potential information piece, or topic, relevant to the patient's presenting complaint. For each topic, the case author defines a set of keywords common in different ways of asking about that topic.

In order to get the best result, LiveBook combines two different interpretation systems. The first system is called Metamap [1] which is a medical thesaurus produced by National Institutes of Health (NIH) in the United States of America. Metamap provides the service of extracting medical terms and their conceptual category from a text. LiveBook using this service extracts medical terms from the learner's question (query). Let's name the set of medical terms in the query Q. Then LiveBook compares Q with keywords of each not investigated topic T with Jaccard distance A.1. The top three topic matches (with lowest Jaccard distance) are then identified as Metamap results. And their distances are normalized among themselves.

The other software used by LiveBook in order to interpret questions is Solr [28]. This software is a text matching software developed by Apache. At the time of creating a case, the topics and their keywords are inserted into Solr. And at the time of interpretation, Solr returns the matched topics with corresponding match score. The same as Metamap's results, the top three matches are selected and their scores are normalized among themselves. Then the topic scores of these six topics are combined and the topic with the highest combined score is the matched topic.

This approach reduces the number of mismatches as Solr would score misspelled words. Also, the authors can fine tune their medical keywords and put more emphasis on a keyword by simply repeating it multiple times. Repeating a keyword increases the score of the topic in Solr match while it does not affect Metamap's score. Also, Metamap's thesaurus is not editable and some words are spelled differently in Canada than in the US, and combining Solr lets the system understand Canadian spelling of the terms too. Also, some concepts have different names in different countries.

It is still possible, however, that LiveBook may misunderstand the learner's question, matching it to an incorrect topic. In these cases, the learner can inform the system about the problem and the problematic question is forwarded

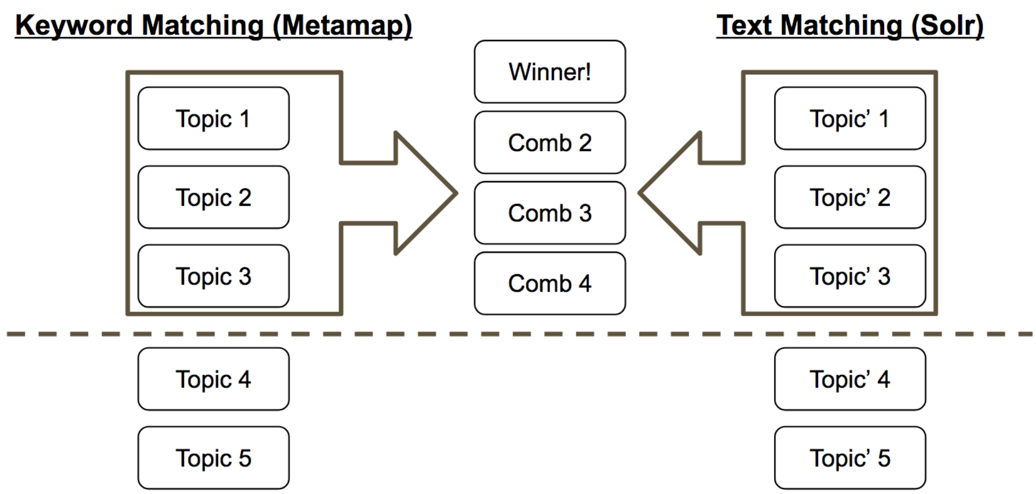


Figure 3.15: LiveBook: Question interpretation, combining the results of Metamap and Solr

to the case author who can tune the keywords for that topic.

3.7 Speech Recognition

One of the main features of LiveBook is its understanding of natural language. It makes the user experience close to real life experience. We went one step further and added a speech-to-text recognition system. The learner can speak his/her question instead of typing it. There is a microphone icon in the question input, the learner can press the microphone button and then start speaking to the device's microphone. Then as soon as the question is over, the learner clicks on the microphone icon again and the recording will be stopped. The system then sends the recorded audio to the server. Next, the server uses a cloud-based speech-to-text service in order to process the question. After the asked question is converted into text, the normal process executes where the user query is substituted with the resulting text. When the response to the question is provided, the learner should check out if the question is understood.

3.7.1 Speech to Text Services

There are different options for services that provide speech to text capability. These options could be either a speech to text service that is installed and executed locally on LiveBook's server or it could be a web-based service. Each has its own advantages. Following comes some of the advantages of the locally executed service:

- There is no limit on the amount of voice that could be converted to text. The only limit is going to be the hardware processing power which could be solved by increasing processing power since the LiveBook servers are based in a cloud.
- The cloud-based speech-to-text services have a fixed vocabulary. If the technical words are not included in the service's vocabulary, we cannot learn the model to understand that word too. But if the service is implemented locally, with some of the packages, the system can also be trained to understand new words.
- The audio file does not need to be uploaded to another server in order to be processed which will end up in providing a faster response.

On the other hand, cloud-based systems have their own advantages too:

- There is no need for processing power to be allocated for speech-to-text processing which could be a heavy process.
- Most of the could-based solutions for this provide the service for free for decent number minutes. We should not exceed the limit in development and in our pilot studies.
- Using cloud-based solutions do not need much installation overhead and it does not add the up-time overhead.

In addition to the technical issues above, an important decision making factor is the result accuracy. Accuracy itself is different based on the microphone, the environment, level of noise, and the speaker's accent. Also, the amount of training data for the model and its coverage of different microphones, environments, levels of noise, and accents is an important factor in the final result accuracy.

We have tested two cloud-based solutions, Google's could-based speech API, and IBM's Watson speech to text service. We have evaluated them using a Mac-book pro with its default microphone in a lab environment and in a normal afternoon with people in the lab. The level of noise was low. We have tested with both men and women, both Canadian born and Immigrant with an accent. Based on this setup they both worked almost perfect. In another experiment, we have replaced our microphone with a headset's microphone connected via Bluetooth. Unfortunately, neither could recognize most of the sentences for any of the speakers. We also evaluated their vocabulary coverage, and they both covered the medical words present in our cases. Even they could understand the abbreviations in the middle of the sentence.

Based on these experiments, cloud solutions were good enough solutions for us. None showed a meaningful advantage over the other. Hence, we provided the capability to use both of them, and the system administrator can switch systems based on whether or not they have reached their usage limit.

3.7.2 Speech to Text Speed

It is well known that raw audio and video files are space inefficient. In our experiment, a normal question asked by the learner will generate a wave file (WAV) between 0.2MB and 1MB. This means that if the internet connection of the learner is no good, it might take some time for the audio file to be uploaded. I have tried some of the audio compression libraries to be installed on the client side. Unfortunately, I was not successful to convert the WAV file into MP3 in order to reduce the size of the audio file. In case it was successful a 221KB file would be converted to 76KB. Although this might improve the service speed, converting to MP3 is going to reduce the audio quality as MP3 is a lossy format and ultimately reduce the speech-to-text accuracy. Another option was to convert WAV to FLAC which is a lossless format. Unfortunately, I could not find a good library to convert WAV to FLAC which would reduce the size from 221KB a 122KB.

On the other hand, the audio file should be uploaded to cloud. Fortunately, both Google and IBM require FLAC format. In LiveBook we convert the WAV file sent to the server to FLAC and then send it to the cloud. Also, as the questions are short, and the overhead required, I did not develop a streaming service which would send the voice straight to the cloud service and fill out the input. This option would reduce the response time as the raw file would be uploaded only once and the learner would see the text before submitting the question and potentially fix the mistakes.

Another factor deciding the size of the audio file is the microphone type. Some microphones record the audio through multiple channels. This will result in a bigger audio file to be generated. For example, if there are two channels which are usually the case for stereo microphones, the generated file would be as twice as big. According to our tests converting stereo and multiple channel audio to mono-track files does not reduce the accuracy in a meaningful way. Therefore, we convert our WAV files in the client JavaScript to mono in order to reduce the size of the file. We do this by averaging the audio samples of all channels.

3.7.3 Browser Compatibility

There is an issue when it comes to recording audio with different browsers and operating systems. This issue has been out there for a long time as part of a bigger issue which is accessing device features from JavaScript in the browser. One of the attempts to resolve this issue is HTML5 where it provides a function getUserMedia() which can be used in order to access user-media resources. Unfortunately, some of the older browsers do not support this feature. In addition, some of the browsers disable access to certain hardware by default. For example, Chrome requires a flag to be updated in order to provide access to the microphone via getUserMedia() even with the user's consent. I could make this feature work in Chrome by configuring the flags. But unfortunately, we could no make this work in Safari. Even Google's search page does not provide speech search through Safari. I hope this issue is resolved in future.

Chapter 4

Simulation-Log Analysis

In standardized-patient simulations with actors, the responsibility of assessing the learner's competence is up to the instructor observing the simulation. Similarly, it is up to the instructor to evaluate the learner's competence, based on the LiveBook's simulation record. The system generates a detailed report on the learner's questions, the specific formulations of these questions and their order, her differential diagnosis at different phases, and her final diagnosis and management plan. To help the instructor with the student-assessment task, LiveBook analyzes the simulation log to generate a detailed report of the learner's performance, as shown in Figure 3.10.

4.1 Simulation-Log Format

LiveBook records the learner activities in order to generate a detailed report. This data is stored in a database with the format shown in Figure 4.1. In this figure, the "Case Study Record" class is created each time a new patient is simulated. Then on each interaction, one "Activity Record" is created based on the type of interaction. The system records asked questions, differential diagnoses, and the management strategy submitted. For each question asked, the matched topic is stored. For each differential diagnosis, the findings summary is stored alongside the diagnosis hypotheses. For each hypothesis, the pertinent information and its rank is also stored. And for each type of management, the form filled for that strategy is stored. This full logging will allow LiveBook to generate valuable feedback to the learner.

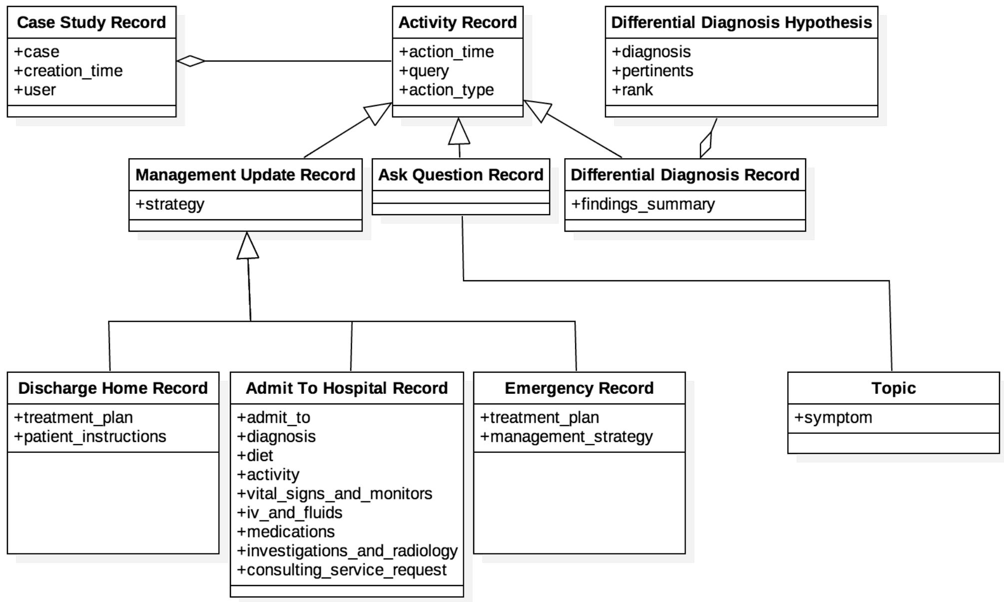


Figure 4.1: Simulation log DB schema

Diagnosis Correctness

In LiveBook, each case pathway has one correct diagnosis, while the learner can provide up to three possible diagnoses. Clearly only one of them can be the correct diagnosis, and if the learner has provided more than one, all but one of them will be wrong. LiveBook records all provided diagnoses, and it states the correct diagnosis. In this manner, the learner and the instructor can easily perceive (a) if incorrect diagnoses have been mentioned, and (b) if so, whether they were the learner's "most likely diagnosis". This analysis provides an indicator of the learner's overall diagnostic competence. This analysis is done manually by the instructor as we could not match the defined diagnosis with the correct diagnosis accurately. The reason for this is that there is no standard format for a diagnosis. For example, "Acute Otitis Media with perforated tympanic membrane" and "Severe Acute Otitis Media in a patient with a suspected Type 1 hypsersensitivity reaction to Beta Lactam antibiotics" are two different diagnoses.

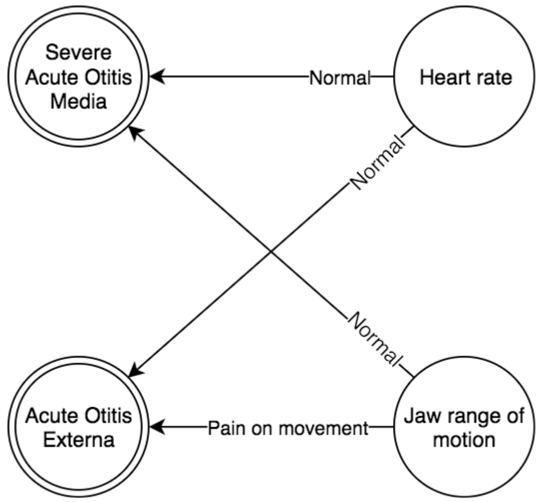


Figure 4.2: Disease-Symptom Relationship Network

4.2 Medical-Knowledge Competence

An important aspect of the learner's knowledge that LiveBook analyzes his/her knowledge of disease/symptom associations. For each diagnosis that the learner provides, LiveBook requires her to also enter the pertinent evidence for it. This is done in two different ways. In the easy version of simulation, LiveBook provides all the gathered evidence from the asked questions and requires the learner to select the pieces of information required to make the diagnosis (Figure 4.3). In the difficult version, LiveBook provides an open text field where the learner types the evidence supporting the diagnosis. Through the easy version this analysis is done automatically and in the difficult version, this is delegated to the instructor.

Consider a situation where all potential diagnoses can be ruled out based on the selected supporting evidence, except the two in Figure 4.2, and the correct diagnosis is "Acute Otitis Externa", which the learner has chosen as the most likely diagnosis. If the learner does not include the "Jaw range of motion" as pertinent evidence, even though she has asked about it, then this symptom will be highlighted yellow. Although the diagnosis is right, the learner should have mentioned all the relevant supporting evidence for her diagnosis.

Finally, if the learner has not elicited a pertinent piece of information, LiveBook simply adds it to the list of pertinent evidence. For example, in the

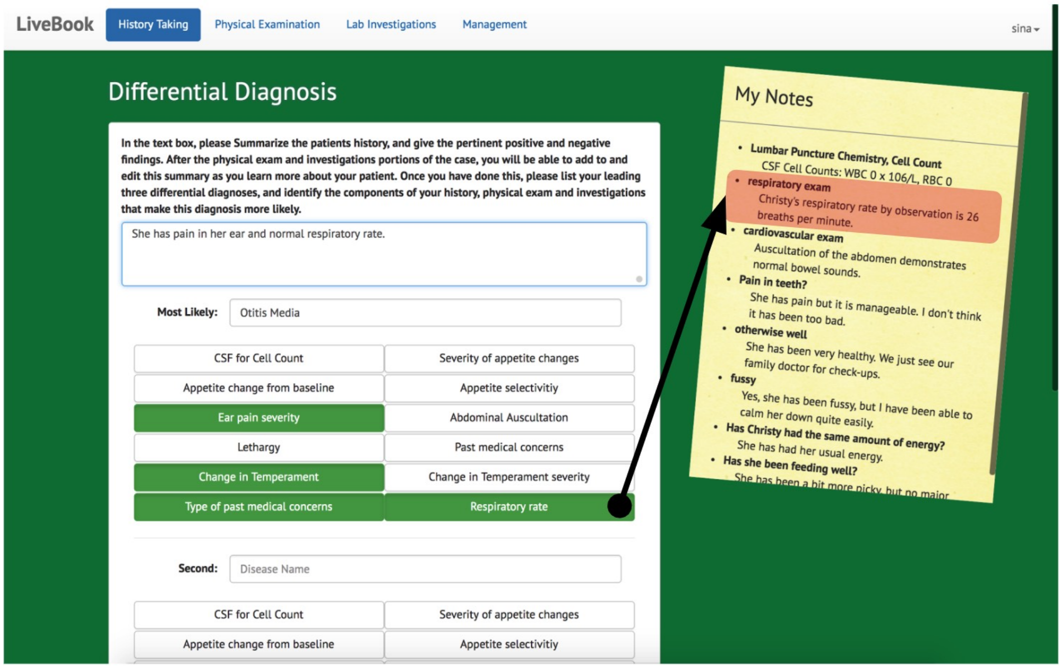


Figure 4.3: LiveBook: Differential diagnosis with supporting pertinent

above scenario, if the learner had not investigated the "Jaw range of motion" of the patient, LiveBook would have mentioned that the learner needed to investigate that too in order to increase her confidence in her diagnosis.

4.3 Diagnostic-Process Skills

To gather information for the purpose of making a diagnosis, the physician has to ask the right questions, namely the questions whose answers may differentiate between alternative candidate hypotheses. Simply collecting additional evidence for a candidate hypothesis is not as good as collecting evidence that advances one hypothesis and, at the same time, eliminates others. LiveBook scores each question, based on how much the question differentiates among potential diagnoses. In that case, the optimal question would be the one expected to reduce the number of possible diagnoses the most.

To formulate LiveBook's scoring system, we defined the score based on the concept of Information Gain, formulating the problem as a decision tree. In the beginning, all diagnoses are possible, as the leaves of the subtree rooted at the root node, with no questions asked yet. By asking a question, the learner moves to a child node, and the subtree rooted at this node includes a subset of the diagnoses of the parent node. The problem then becomes to find the best question to ask at the current node, which will produce the highest information-gain value. We define S as the set of potential diagnoses.

$$S = \{d \mid d\, is\, possible\} \tag{4.1}$$

Then, the entropy at a node can be computed by the following formula, assuming equal probability for potential diagnoses.

$$Entropy(S) = -\sum_{i=1}^{|S|} P(d_i) log(P(d_i)) \tag{4.2}$$

Assuming that different diagnosis have the same probability, the probability of each diagnosis will be:

$$P(d_i) = \frac{1}{|S|} \tag{4.3}$$

It is worth mentioning that this means that the system ignores the general statistics and epidemics. For example, the information that Otitis is a common disease in children, it is not considered as a factor here. Likewise, if there has been a breakout of a disease, it is not considered as an effective factor in the objective reasoning for the diagnosis.

With this assumption and by substituting 4.3 in 4.2 we get:

$$Entropy(S) = -\sum_{i=1}^{|S|} \frac{1}{|S|} log(\frac{1}{|S|}) = -log(\frac{1}{|S|}) \tag{4.4}$$

Then the entropy after the question Q has been asked can be computed by the equation below, assuming there are k different outcomes for Q (k children nodes), and S_i is the set of potential diagnoses for the ith child node.

$$Entropy(Q, S) = \sum_{i=1}^{k} Entropy(S_i) \tag{4.5}$$

Finally, the information gain can be computed as:

$$\begin{aligned} Gain(Q,S) &= Entropy(S) - Entropy(Q,S) \\ &= -log(\frac{1}{|S|}) - \sum_{i=1}^{k} log(\frac{1}{|S_i|}). \end{aligned} \tag{4.6}$$

In our example in Figure 4.2, S would be the set of "Severe Acute Otitis Media" and "Acute Otitis Externa". At this point the entropy would be $-log(\frac{1}{2}) = 1$. If the learner asks about "Heart rate", the only response is "Normal" and the entropy after this question would be the same as before, and the information gain would be 0. On the other hand, if the learner asks about "Jaw range of motion", the entropy after the question would be 0 and the information gain would be 1. So the optimal question is "Jaw range of motion" which decides the diagnoses. If the learner asks about "Heart rate" the information gain is 0 as it could not possibly rule out any diagnosis and would be considered as an irrelevant question.

At each stage, the learner can ask about anything including the already investigated topics. In case a topic is investigated again, it will have 0 as Information Gain value. Let Q^* be the optimal question with the highest information gain among all topics. LiveBook determines the score of the question Q, based on the following formula:

$$Score(Q,S) = \frac{Gain(Q,S)}{Gain(Q^*,S)},\ (|S| > 0) \tag{4.7}$$

This score is going to be between 0 and 1. It is going to be 1 when the learner asks the optimal question and 0 when the asked question cannot potentially eliminate any candidate diagnosis.

Chapter 5

Deployment Study

We have attempted to empirically evaluate LiveBook. We recruited residents from the general pediatrics program at the University of Alberta, and content experts in the field of Pediatric Infectious Diseases at the University of Alberta. In collaboration with these experts, we have developed a virtual-patient case with 16 pathways around Acute Otitis Media. The case consists of 82 topics, each one associated with 2 to 16 medical keywords. Through this pilot, we expected to obtain the case-specification data and much constructive feedback regarding the overall case design and learning outcomes.

We recruited 7 participants from the programs mentioned above. Participation was voluntary and non-compensated. All eligible residents were invited to participate, and the only inclusion criterion was registration within a general pediatrics training program in Canada. Only individuals who did not consent to participate in the trial or were case authors were excluded from participation.

The participants were given an ID which they could use to signup into LiveBook. They were asked to go through a case in LiveBook using their laptop or mobile at their own convenient time. The participants went through the cases at different times throughout a day from 8 a.m. to 10 p.m. We did not track what device type our participants used in order to go through the cases.

Learning outcomes were expected to be assessed using pre and post-testing and student satisfaction surveys (both qualitative and quantitative), and by monitoring the manner in which students and experts complete each case (time spent, number and organization of questions, physical exam maneuvers, and investigations used, number of times the learner went through the case, stated differential diagnosis, pertinent positives and negatives, etc.). How much the experience were realistic has be assessed using the EVip published evaluation instrument for Virtual Patients ([20], [13]).

Due to technical issues, the pre-test surveys were not recorded. We have gathered the remaining data from this pilot and analyzed it. Each participant was asked an EVip survey after the study and each question had the following options; NA=Not applicable, 1=Strongly Disagree, 2=Disagree, 3=Neither agree/nor disagree, 4=Agree, and 5=Strongly agree. Participant number 2 did not participate in the EVip survey. The results of the survey are shown in Table 5.1 and the questions are as follows:

1. While working on this case, I felt I had to make the same decisions a doctor would make in real life.

2. While working on this case, I felt like I was the doctor caring for this patient.

3. While working through this case, I was actively engaged in gathering the information (e.g., history questions, physical exams, lab tests) I needed to characterize the patient's problem.

4. While working through this case, I was actively engaged in revising my initial interpretation of the patient's problem as new information became available.

5. While working through this case, I was actively engaged in creating a short summary of the patient's problem using medical terms.

6. While working through this case, I was actively engaged in thinking about which findings supported or refuted each diagnosis in my differential diagnosis.

Table 5.1: LiveBook: EVip questions survey result (from Strongly disagree to Strongly agree; from 1 to 5)

Question Num.	**P1**	**P3**	**P4**	**P5**	**P6**	**P7**
Q 1	4	5	5	4	5	5
Q 2	3	5	4	4	4	3
Q 3	3	5	4	4	5	4
Q 4	4	5	4	4	5	4
Q 5	3	5	5	4	5	3
Q 6	4	5	5	4	5	4
Q 7	4	4	5	4	4	4
Q 8	3	5	5	4	4	4
Q 9	4	5	5	N/A	3	4
Q 10	4	5	4	3	4	4
Q 11	3	5	4	4	3	4
Q 12	3	5	5	4	4	4

7. I felt that the case was at the appropriate level of difficulty for my level of training.

8. The questions I was asked while working through this case were helpful in enhancing my diagnostic reasoning in this case.

9. The feedback I received was helpful in enhancing my diagnostic reasoning in this case.

10. After completing this case, I feel better prepared to confirm a diagnosis and exclude differential diagnoses in a real life patient with this complaint.

11. After completing this case I feel better prepared to care for a real life patient with this complaint.

12. Overall, working through this case was a worthwhile learning experience.

The results in Table 5.1 show that LiveBook was successful in simulating the role of the doctor and engaging the participants. In each question, the majority of the participants responded that they agree or strongly agree. The

Table 5.2: LiveBook: Experiment Results.

ID	# of Questions	Case Time (min)	Avg. Question Score	Avg. Question Time (sec)	Pathway
1	25	9	0.40	14	C1
2	14	10	0.49	39	A2
3	39	31	0.46	11	B2
4	41	24 (91)	0.55	23	B3
5	20	11	0.66	11	C2
6	36	20	0.46	13	B3
7	11	14	0.61	17	A3

Table 5.3: LiveBook: Experiment time allocation results.

ID	Case Time (min)	Differential Diagnosis & Management Time (min)	Questions Time (min)
1	9	3	6
2	10	1	9
3	31	24	7
4	24	9	16
5	11	8	4
6	20	12	8
7	14	11	3

analysis of this survey also shows that participants who asked more questions and the ones that spent more time using the system were more satisfied with their experience.

The results of the simulation are shown in Table 5.2 including the number of asked questions, the time spent on the case, the average score for each asked question, the average time spent on each asked question, and the case they studied (pathway). The time split between asking questions and filling the differential diagnosis and the management strategy is also shown in Table 5.3.

It is interesting that our participants have spent 3 to 16 minutes asking questions. Participants number 4 and 6 had the same case, but one of them asked 16 minutes of questions in order to make a diagnosis while the other spent half the time to come up with a diagnosis. On average our participants spent seven and a half minute asking questions in the simulation.

Our participants have asked from 11 to 41 questions. Clearly, they did the diagnosis in different manners as a participant asked almost four times questions than another. They have asked on average 27 questions.

The amount of time spent on the cases are also very different. They have spent 9 to 31 minutes (more than three times). One of the participants (number 4) stopped the simulation for more than an hour and came back. Being able to use a simulation and having the freedom of using the system at their own time of convenience was a helpful feature for this participant.

The question scores average from 0.4 to 0.66. There is no correlation between the number of asked questions and their score. The two participants who had the same case got different question scores of 0.55 and 0.46. Although they both asked a lot of questions (41 and 36), LiveBook gave them different scores for their asked questions.

Each participant has spent on average 18 seconds to ask a question. This shows that they did not have a lot of trouble asking their questions. Considering the time between asked questions includes the student reading the response and thinking about asking another question, the average 18 seconds is not a high value. From over 200 asked questions in the simulation 28 of them were reported as misunderstood. This means about one out of seven questions are reported as misunderstood. Meanwhile, the medical members of our team went through cases without any misunderstood question. This means that after using the system for a while the students can adapt themselves with how to ask about something they are looking for.

Chapter 6

Conclusion

In this thesis, we described LiveBook, an interactive simulation system that makes three key contributions to the state of the art. First, it supports a natural interaction model, by understanding natural-language spoken questions and responding with answers also in natural language. LiveBook's question interpretation system recognizes the learner's question as one of the questions it knows about, based on their overall-term and medical-keywords similarity. It also includes a mechanism to expand its question knowledge base, enabling the learner to indicate that her question was not properly understood. Second, through its case-pathway generation process, it is able to offer multiple learning experiences based on each case specification, thus enabling learners to spend more time on the task and practice their knowledge and skills longer. Finally, LiveBook includes a simulation-log analysis functionality; the reports generated by the system are annotated with LiveBook's evaluation of the correctness of the student's diagnosis, the accuracy and completeness of his/her disease-symptom knowledge, and the quality (from an information-gain perspective) of his/her questions. Based on this annotated log, instructors can give more specific and constructive feedback to the students, a key enabler of effective learning [22].

In simulation-based medical education different factors contribute to the learner's overall improvement. Feedback, deliberate practice, and simulation fidelity are among these features and the most important factor is constructive feedback [22]. LiveBook provides detailed simulation log, in CSV and report

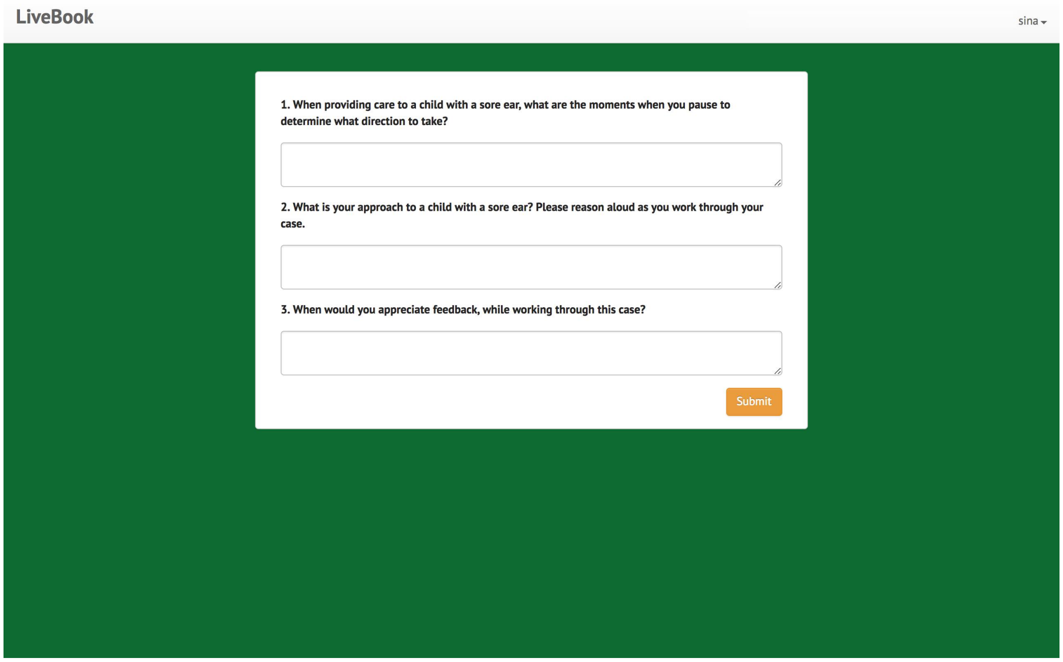


Figure 6.1: LiveBook: Participation survey

card format, for the instructor in order to enable them to evaluate the learner's performance and provide feedback. On the other hand, medical instructors' time is valuable and expensive and an automated feedback system would help the learners to improve their performance. Because the learner is no longer dependent on the instructor to access constructive feedback, he can go through a lot more cases and gain more experience.

Web-based systems have the advantage of being accessible from almost everywhere, at any time, and with any device that connects to the Internet. We have developed LiveBook as a web-based system and mobile friendly to make it as accessible as possible.

The medical members of our team have used and tested the system and they appreciate the usability of the interface and its fidelity to real-life clinical reasoning. They could easily go through a diagnosis problem and the system understood their questions correctly almost all the time. They were also happy with the user interface and did not have any problem traversing the website and asking questions. LiveBook can facilitate clinical reasoning as the students will experience of coming up with a diagnosis and a management strategy, but

LiveBook does not provide any feedback on them except for a comparison with an expert. Thus, LiveBook has the potential to educate the student in clinical reasoning but it does not do it now.

JSON is a simple format, which makes creating new cases and editing older ones easy. The case-specification language minimizes case content by removing all duplicate information in JSON format. LiveBook also provides a user-friendly interface for developing cases which reduces the time required to make a case by filling the form fields automatically.

6.1 Statement of Contributions and Lessons Learned

LiveBook provides two main systems, Quiz and Case Simulation. The quiz module generates quizzes automatically based on the incrementally expanded medical database. It enables the instructors to generate quizzes and evaluate the students based on their performance. The case simulation module enables the instructors to author and edit cases using a user-friendly interface. Then it allows the students to go through the cases simulating the role of the doctor for the students to create a realistic experience where the student can speak questions or order lab tests and get the response in natural language along media contents. The contributions of LiveBook is as follows:

- LiveBook's services are based on a Medical Knowledge Base (MKB). This enables the system to improve by adding more data to the MKB. As more data is available, it can generate better quizzes, and improve its feedback, and provide better help in case generation process.
- LiveBook generates quizzes automatically based on its MKB.
- Instructors can develop and edit cases using a user-friendly interface where they are provided assistance from the system via auto-completing the fields.
- Students can run simulations as they wish over the web where they can speak or type their questions using natural language. The response is also

provided in natural language format and the rare case that the system cannot understand the question it will ask the student to rephrase the question.

- LiveBook is created in modular format; therefore, expanding the system is easy. This is true also for the medical knowledge base. The MKB is created as it would be easily expandable.

LiveBook provides four out of six learning levels from Bloom's Taxonomy; remember, understand, apply, analyze. The evaluation and synthesis levels are also experienced by the students when they run a simulation, but LiveBook does not provide valuable feedback on them. Providing the feedback on these two levels would be difficult. LiveBook needs to make sure that it can match the provided diagnosis with one of the diagnoses it knows about. That matching cannot be done correctly all the time as it is a string representation of the diagnosis which might be matched with a wrong diagnosis. LiveBook would also need to provide feedback on the supporting evidence of the diagnosis and it cannot over all the information provided through the text as well.

During development of LiveBook, we learned technical points on how to make the system flexible and expandable. As previous versions of LiveBook did not have an MKB, it would not be possible to insert new data into the database by merging new data from other databases. The modular architecture of LiveBook enables the system to develop mobile apps, add new systems, and run the system on the cloud with parallel machines. We also learned how to make the experience more realistic and make the system interface more user-friendly. We had multiple back-and-forths with the medical members of our team as they would provide comments from the user perspective. We also learned about the importance of testing as we lost the data of our experiment. All the work that has been put on the experiment was lost due to our mistake.

6.2 Future Work

LiveBook currently includes a question interpretation system. But it currently cannot detect misspelled words. We can try to find a potentially correct spelling in case the question cannot be understood by the system.

Another area for improvement is adding a text-to-speech module where the response of the patient is also played for the user. This will make the experience even more realistic. The system is currently modular and can be easily extendable. One way of extending the current system is to make the phases customized. In that case, LiveBook becomes an across disciplines software which can be used in different fields.

LiveBook generates quizzes and provides auto-completion in generating cases based on its Medical Knowledge Base (MKB). Expanding this database is currently done through generating new cases. But we can develop a tool where doctors can contribute to this database.

LiveBook evaluates a learner's performance in asking valuable questions based on its MKB. LiveBook can also generate cases based on its data given that the database is also filled with data other than the current cases. The only blocker for LiveBook would be its capability of generating natural language responses based on the case.

References

[1] A. R. Aronson, "Effective mapping of biomedical text to the umls metathesaurus: The metamap program.," in *Proceedings of the AMIA Symposium*, American Medical Informatics Association, 2001, p. 17. 34

[2] J. Bateman, M. Allen, D. Samani, J. Kidd, and D. Davies, "Virtual patient design: Exploring what works and why. a grounded theory study," *Medical education*, vol. 47, no. 6, pp. 595–606, 2013. 5, 64

[3] R. A. Bergin and U. G. Fors, "Interactive simulated patient—an advanced tool for student-activated learning in medicine and healthcare," *Computers & education*, vol. 40, no. 4, pp. 361–376, 2003. 73

[4] N. B. Berman, S. J. Durning, M. R. Fischer, S. Huwendiek, and M. M. Triola, "The role for virtual patients in the future of medical education.," *Academic medicine: journal of the Association of American Medical Colleges*, 2016. 6

[5] A. M. Burrows, W. V. Norman, and L. M. Currie, "Learners' critical thinking processes when using virtual patient cases," in *NI 2012: Proceedings of the 11th International Congress on Nursing Informatics*, American Medical Informatics Association, vol. 2012, 2012. 6

[6] D. Chodos, E. Stroulia, and P. Naeimi, "An integrated framework for simulation-based training on video and in a virtual world," *Journal For Virtual Worlds Research*, vol. 2, no. 1, 2009. 62

[7] D. A. Cook, P. J. Erwin, and M. M. Triola, "Computerized virtual patients in health professions education: A systematic review and meta-analysis," *Academic Medicine*, vol. 85, no. 10, pp. 1589–1602, 2010. 7

[8] F. E. Craig, J. B. McGee, J. F. Mahoney, and C. G. Roth, "The virtual pathology instructor: A medical student teaching tool developed using patient simulator software," *Human pathology*, vol. 45, no. 10, pp. 1985–1994, 2014. 63

[9] R. Ellaway, C. Candler, P. Greene, and V. Smothers, "An architectural model for medbiquitous virtual patients," *Baltimore, MD: MedBiquitous*, 2006. 6

[10] E. Forsberg, C. Georg, K. Ziegert, and U. Fors, "Virtual patients for assessment of clinical reasoning in nursing—a pilot study," *Nurse Education Today*, vol. 31, no. 8, pp. 757–762, 2011. 11, 69

[11] B. T. Harris, D. Montero, G. T. Grant, D. Morton, D. R. Llop, and W.-S. Lin, "Creation of a 3-dimensional virtual dental patient for computer-guided surgery and cad-cam interim complete removable and fixed dental prostheses: A clinical report," *The Journal of prosthetic dentistry*, vol. 117, no. 2, pp. 197–204, 2017. 72

[12] G. Huang, R. Reynolds, and C. Candler, "Virtual patient simulation at us and canadian medical schools," *Academic Medicine*, vol. 82, no. 5, pp. 446–451, 2007. 5

[13] S. Huwendiek, B. A. De Leng, A. A. Kononowicz, R. Kunzmann, A. M. Muijtjens, C. P. Van Der Vleuten, G. F. Hoffmann, B. Tönshoff, and D. H. Dolmans, "Exploring the validity and reliability of a questionnaire for evaluating virtual patient design with a special emphasis on fostering clinical reasoning," *Medical teacher*, vol. 37, no. 8, pp. 775–782, 2015. 47

[14] G. Isaacs, "Bloom's taxonomy of educational objectives," *Teaching and Educational Development Institute. Queensland (Australia): University of Queensland*, 1996. 2

[15] P. Jaccard, "The distribution of the flora in the alpine zone.," *New phytologist*, vol. 11, no. 2, pp. 37–50, 1912. 23, 59

[16] M. S. Janda, N. Mattheos, A. Nattestad, A. Wagner, D. Nebel, C. Färbom, D.-H. Lê, and R. Attström, "Simulation of patient encounters using a virtual patient in periodontology instruction of dental students: Design, usability, and learning effect in history-taking skills," *European Journal of Dental Education*, vol. 8, no. 3, pp. 111–119, 2004. 65

[17] E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, p. 620, 1957. 59

[18] A. Jena, *Fuseki: Serving rdf data over http*, 2014. 20

[19] A. Kleinheksel, "Transformative learning through virtual patient simulations: Predicting critical student reflections," *Clinical Simulation in Nursing*, vol. 10, no. 6, e301–e308, 2014. 64

[20] B. de Leng, S. Huwendiek, and J. Donkers, "Published evaluation instrument for virtual patients and for learning and teaching activities with virtual patients," 47

[21] W. C. McGaghie, S. B. Issenberg, M. E. R. Cohen, J. H. Barsuk, and D. B. Wayne, "Does simulation-based medical education with deliberate practice yield better results than traditional clinical education? a meta-analytic comparative review of the evidence," *Academic medicine: journal of the Association of American Medical Colleges*, vol. 86, no. 6, p. 706, 2011. 4

[22] W. C. McGaghie, S. B. Issenberg, E. R. Petrusa, and R. J. Scalese, "A critical review of simulation-based medical education research: 2003–2009," *Medical education*, vol. 44, no. 1, pp. 50–63, 2010. 4, 5, 51

[23] ——, “Revisiting ‘a critical review of simulation-based medical education research: 2003–2009’,” *Medical education*, vol. 50, no. 10, pp. 986–991, 2016. 4, 5

[24] E. Menendez, B. Balisa-Rocha, M. Jabbur-Lopes, W. Costa, J. R. Nascimento, M. Dósea, L. Silva, and D. L. Junior, “Using a virtual patient system for the teaching of pharmaceutical care,” *International journal of medical informatics*, vol. 84, no. 9, pp. 640–646, 2015. 67

[25] Y. Okuda, E. O. Bryson, S. DeMaria, L. Jacobson, J. Quinones, B. Shen, and A. I. Levine, “The utility of simulation in medical education: What is the evidence?” *Mount Sinai Journal of Medicine: A Journal of Translational and Personalized Medicine*, vol. 76, no. 4, pp. 330–343, 2009. 1

[26] N. Posel, J. B. Mcgee, and D. M. Fleiszer, “Twelve tips to support the development of clinical reasoning skills using virtual patient cases,” *Medical teacher*, vol. 37, no. 9, pp. 813–818, 2015. 5, 64

[27] A. Sliney and D. Murphy, “Jdoc: A serious game for medical learning,” in *Advances in Computer-Human Interaction, 2008 First International Conference on*, IEEE, 2008, pp. 131–136. 61

[28] A. Solr, *Apache solr*, 2011. 34

[29] C. Stepniak, B. Wickens, M. Husein, J. Paradis, H. M. Ladak, K. Fung, and S. K. Agrawal, “Blinded randomized controlled study of a web-based otoscopy simulator in undergraduate medical education,” *The Laryngoscope*, vol. 127, no. 6, pp. 1306–1311, 2017. 70

[30] J. Wiecha, R. Heyden, E. Sternthal, and M. Merialdi, “Learning in a virtual world: Experience with using second life for medical education,” *Journal of medical Internet research*, vol. 12, no. 1, 2010. 5

[31] G. L. Wilkening, J. M. Gannon, C. Ross, J. L. Brennan, T. J. Fabian, M. J. Marcsisin, and N. J. Benedict, “Evaluation of branched-narrative virtual patients for interprofessional education of psychiatry residents,” *Academic Psychiatry*, vol. 41, no. 1, pp. 71–75, 2017. 68

[32] N. Zary, G. Johnson, and U. Fors, “Web-based virtual patients in dentistry: Factors influencing the use of cases in the web-sp system,” *European Journal of Dental Education*, vol. 13, no. 1, pp. 2–9, 2009. 70

[33] N. Zary, G. Johnson, J. Boberg, and U. G. Fors, “Development, implementation and pilot evaluation of a web-based virtual patient case simulation environment–web-sp,” *BMC medical education*, vol. 6, no. 1, p. 10, 2006. 10, 69

[34] X. Zhou, J. Menche, A.-L. Barabási, and A. Sharma, “Human symptoms–disease network,” *Nature communications*, vol. 5, 2014. 20

Appendix A

Background Material

A.1 Jaccard Distance

Jaccard distance is a mathematical formula computing a measure of distance between two sets [15]. The formula is the ratio of the number of items in their intersection divided by the number of items in their union.

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{A.1}$$

A.2 Information Theory

Information theory (IT) is a mathematical field for modelling information. It studies how to quantify information, store it, or communicate it [17]. The amount of information is represented by a measure called Entropy. Entropy means the uncertainty or surprise of the result of an event. It assumes that whenever there is no information, any outcome is possible and when there is no more information to be gained, the outcome is determined and there is no surprise.

In order to quantify information, we need a unit like the way we quantify weight, length, and time. The unit in IT is called bit. One bit represents the outcome of a fair coin toss. Considering a source for the information, the entropy of the information produced is calculated by the following formula:

$$H = -\sum_{i \in outcome} p_i log_2(p_i), \tag{A.2}$$

where for each possible outcome, p_i is the probability of that outcome.

A.2.1 Information Gain

Information gain (IG) in probabilistic phenomena, is the difference of the amount of information available before and after an observation. Let O be an observation. The IG for O in the output X is calculated with the following formula:

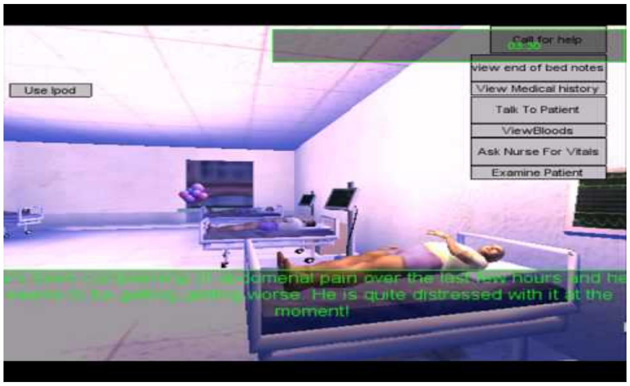
$$IG(O, X) = H(X) - H(O|X) \tag{A.3}$$

For example, if the outcome X represents the number of tails in 2 fair coin flips, without any observation the entropy would be $\frac{3}{2}$. But if we already observed that the first toss resulted in a tail, the entropy would be 1. This means that we now have more information about X after observation O. The amount of information that we gained is $\frac{1}{2}$. The IG of the same observation would be zero if the outcome was whether or not the tosses result in the same face. The probability of the same face showing up is $\frac{1}{2}$ in both cases.

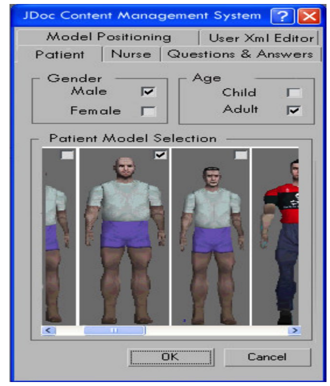
A.2.2 Information Gain and Decision Trees

In our system, we can model the process of diagnosis with a decision tree. Consider a node R to be the root of a tree. Initially, all diagnoses are in the root R. At each none-leaf node, we can ask about a topic. Different answers to our question result in different children of that node. And in each leaf node there is one true diagnosis, assuming each two diagnoses can be distinguished by investigating a topic. By this assumption, we can always find the right diagnosis by asking about all the topics, but the learner is supposed to find the right diagnosis with the minimum number of questions asked. This means that he is trying to reach a leaf node with the lowest number of questions.

There are different trees for each diagnosis and the learner is trying to reach a leaf faster not knowing which tree he is in. At each step, assuming all the questions asked, he should find the topic which is expected to eliminate more remaining trees. As soon as the entropy reaches zero the right diagnosis



(a) A view of the JDoc system



(b) A view of JDoc avatar modeling

is found because given the right diagnosis we know the answer to all topics and there is no uncertainty or surprise left. It is worth mentioning that this model assumes the learner's medical knowledge is complete and he knows all the relationships between the diseases and their symptoms.

Using this model, we can model the diagnosis process with a decision tree and the value of each question in a node with the information gained by asking that question, since the ultimate purpose is to reach entropy zero as soon as possible.

A.3 Related Software

In this section we compare the related software in the field of Simulation-Based Medical Education (SBME) software with LiveBook. We describe each software in a little bit and then illustrate some of the advantages and distadvantages of them compared to LiveBook.

A.3.1 JDoc

JDoc [27] is a game designed for junior doctors to experience a busy and hectic night in a hospital. The user plays the role of a doctor. In each simulation, the user must find the hospital and then find the patient and assess his/her situation. The process includes interacting with other doctors, nurses, and objects. Senior doctors create the scenarios the learner (junior doctor) goes through.

The software is created for desktop computers. It uses Torque Game Engine from GarageGames and the modelling is done by 3DS Max 8 (screenshot in Figure A.1b) and the game engine is created using C++. The game can be played only on desktop computers as it is created for desktop machines which limits the learner to use the system only when they have this game installed on their machine and they can only practice when they have access to their laptops or desktop PCs.

JDoc makes realistic experience by simulating the game in a 3-dimensional environment and enabling the user to interact with people and objects. The interactions are provided in form of multiple options for the user to select from and the response is provided in a visual or audio format or both. The input methods to JDoc is mouse and keyboard, and the JDoc interacts with the user using monitor and speakers. The quality of the audio output helps the environment become closer to what a doctor may experience in a real hospital. One learner can go through multiple scenarios as multiple cases can be created in JDoc. The system is evaluated based on its usability with 28 subjects of mixed sex. The evaluation is done subjectively based on questionnaires after the subject has gone through the game.

At the end of a simulation, JDoc provides the actions of a senior doctor in the same situation. The system itself cannot analyze the learner's performance; it is designed based on the assumption that the learner evaluates his/her performance by comparing their own actions to those of a more senior colleague, junior doctors recognize their strengths and shortcoming and may set objectives for their own improvement. At the same time, instructors can evaluate the learner's actions and performance.

A.3.2 MERiTS

The MERiTS framework [6] proposes the Avatar Capabilities Model (ACM) of the educationally relevant actions that a student can take within a virtual world. Relying on the ACM, an instructor defines a set of non-deterministic workflows representing the normative behaviour of actors under a particular scenario setting. At run-time, as the students are enacting the various sce-

Figure A.1: A view of the MERiTS system

nario roles, their actions are recorded and analyzed against the normative behaviour, as defined by the instructor. These analyses provide instructors with systematically collected evidence of the students' competence and knowledge. MERiTS was evaluated with emergency-room respiratory-problem scenarios and road-side accident responder scenarios.

MERiTS utilizes visual and audio interactions for simulating the situation which provides a more realistic experience. It is a general purpose software as it can be used in different fields of study.

Although MERiTS is more realistic in its representation of the activity setting through a 3D graphical environment, it offers a less than ideal, menu-driven interaction method to the learner. LiveBook's natural-language interaction method enables a more natural experience for the learner.

A.3.3 V-PIN

A number of virtual-patient systems define the learner's environment as a combination of natural-language text and media elements. An example of such systems is the Virtual Pathology Instructor (V-PIN) [8], which simulates the role of a diagnostic pathologist. The learner goes through a list of slides, answering a set of questions on each one. The learner cannot continue before they produce the right answer, eventually reaching a final diagnosis.

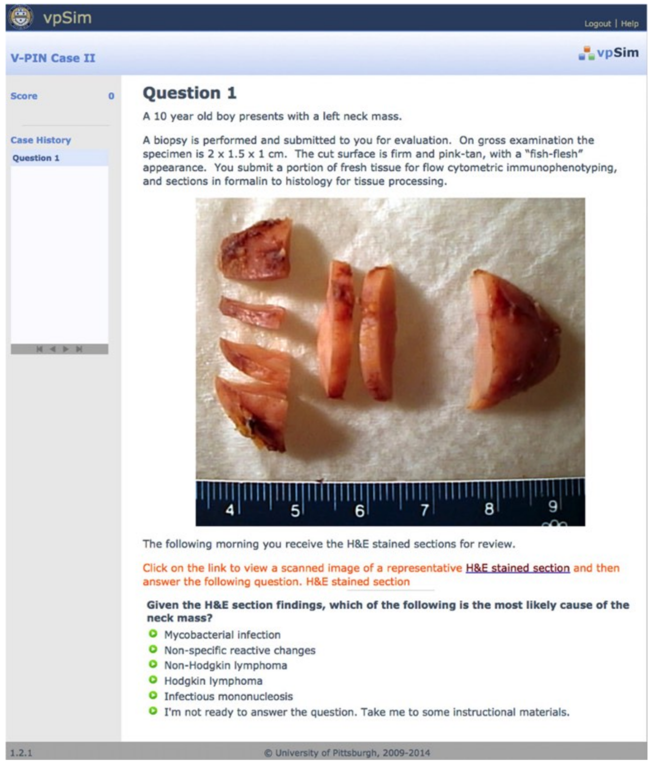


Figure A.2: A view of V-PIN system

V-PIN walks the learner through the process that a senior doctor would go through. This enables the learners to compare themselves to an expert and understand when they make a wrong choice. It also enhances the experience using media content which can illustrate real-world information.

This software walks the learner through a case with the correct answer in each step. In contrast, LiveBook allows the learner to explore the case, asking any question they wish, including possibly irrelevant ones, and to document their diagnoses without insisting that they have the correct one. This is a more realistic, and, we believe, pedagogically useful, experience ([2] and [26]) as it provides branched scenarios.

A.3.4 Digital Clinical Experience

Similar to MERiTS in Section A.3.2, the Digital Clinical Experience (DCE) [19] simulator places the learner, in the role of a nurse, in a virtual hospital room where she interviews and examines the patient, using a number of medical devices. In the end, the student writes a self-assessment report and the

Figure A.3: A view of the Digital Clinical Experience system

system provides a scorecard containing the important points that the learner may have missed, by comparing the student's interaction with the patient's avatar and devices against a predefined list of desired interactions.

The visualization and virtual interactions make the experience more realistic as it makes it more enjoyable. The report card enables the learner to understand what could he/she investigate that was missing.

LiveBook adopts a natural-language interaction model with the learner, but its case-specification language enables it to interpret every exchange (student questions and patient's answers) in terms of the information they communicate about the presence or absence of different symptoms. In this manner, it is able to better evaluate the student's competence because it can consider not only each individual question but also how the student's questions together lead to his/her diagnosis. LiveBook also provides a score for each individual question representing how helpful that question was in the process.

A.3.5 VLE for oral health-care

Janda et. al[16] developed a virtual learning environment (VLE) for teaching periodontology. The system enables the learner to interact with the patient using natural language and provides text and media elements in the response.

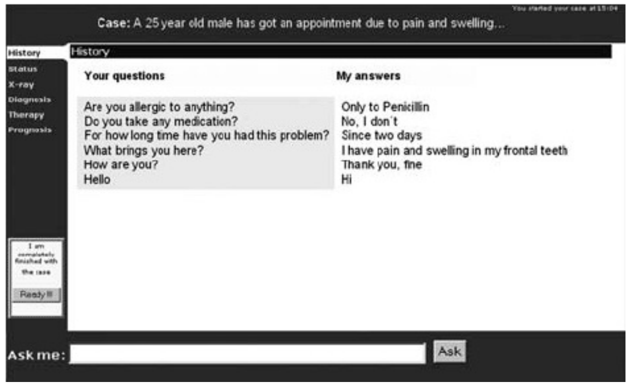


Figure A.4: A view of the VLE for oral health-care

The learner asks questions and the system responds based on a case definition in the database. The process itself is split into some sections like history, X-ray, therapy, and diagnosis.

This is a web-based service that records the learner's activities and generates reports for the instructor. It includes the asked questions, images shown to the learner, time spent on each question, and movements between sections. This will enable the instructor to provide valuable feedback to the learner. It also sends a list of questions which the system failed to find in the case to the instructor in order to improve the case for the future studies. The learner goes through a realistic experience in the role of the doctor and is provided natural language interaction with media elements as required.

However, unlike LiveBook, it does not support branching or alternative pathways through a case, which is a key attribute of our system's realism. This LiveBook feature provides the learner with the opportunity to detect different characteristics of different diagnoses, which is an essential skill for diagnostic problem solving and clinical reasoning. LiveBook also provides automated feedback on the asked questions and the diagnosis. This will save valuable time since the learner can go through multiple cases without needing the instructor to provide feedback on each case.

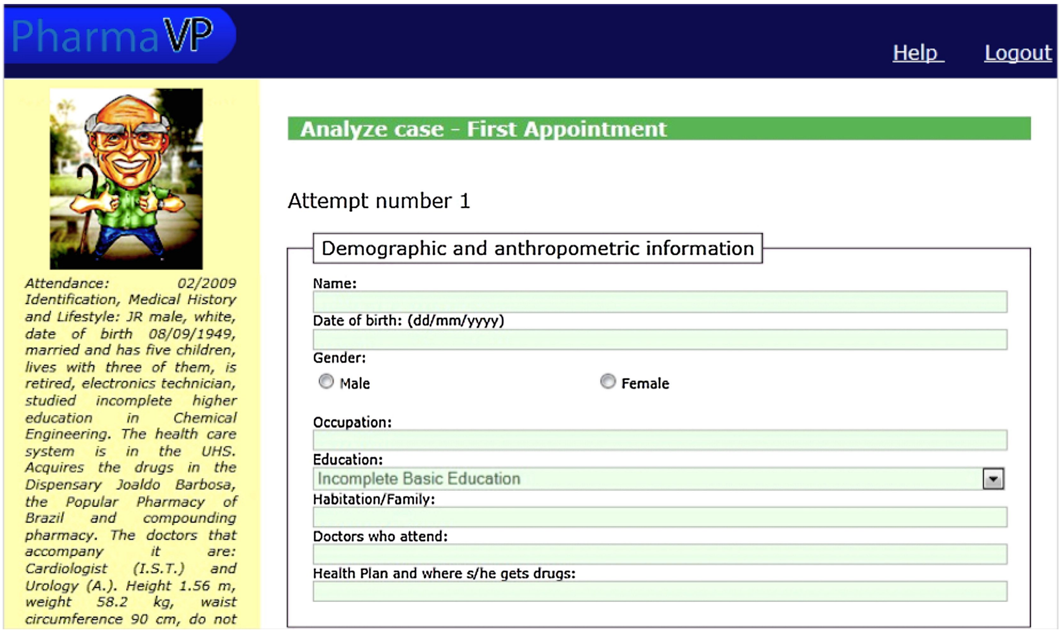


Figure A.5: A view of the PharmaVP system

A.3.6 PharmaVP

PharmaVP was developed for training pharmacy students, by simulating different visits of a patient to a pharmacy, and also for evaluating the trainer's skills [24]. In this system, students have to fill out a case report that has to be evaluated by the tutor before they can proceed to the next step, which may involve the next visit of the virtual patient for the same or a new case.

This method helps the learner experience multiple visits of a patient with potentially different cases. The learner is put in the role of the doctor and he/she will receive feedback from an expert after each visit. PharmaVP is developed as a web application which enables the learner to study from anywhere with internet access.

LiveBook allows learners to run as many simulations as they wish, without requiring the feedback of the instructor between them, thus allowing for asynchronous evaluation and enabling the use of the system whenever convenient for the learner. This is one of the PharmVP deficiencies which we overcome in LiveBook by adding the evaluation module to our system in order not to take instructor's time for every student going through a case.

A.3.7 Branched-Narrative Simulator based on Dicision-Sim

In their paper [31], authors deployed a branch-narrative virtual patient system where the students are given some choices based on a challenge, then the system shows them the consequences of their choices. Therefore, the success of the learner is defined by his or her recommendations in aggregate, because, each recommendation option relates to a branch of the clinical scenario. The system interacts with learners through text, graphics, and video. To create each virtual patient scenario, psychiatric pharmacist and psychiatrists rotation directors meet to discuss psychopharmacology topics, and the ones which are needed more for the learning purpose is selected. During each scenario, students are first presented with a challenge, they provide some treatment plan. Then based on their plan, the system gives them negative or positive outcomes on the virtual patient in addition to the feedback developed by clinical experts. Based on these feedbacks, the students can alter their responses. To evaluate the system's performance, learners complete two online pre-assessment and post-assessment for each virtual patient. The assessment questions were designed by the psychiatric pharmacists. Although this system provides some feedback, they are limited to the outcome of the learner's decisions on the patient, and the instructor's feedback is still crucial for validating the student's answer, same as PharmaVP.

This system provides an interactive environment for the learner with branching narrative where the learners experience the results of their actions. This will enable them to make mistakes in the virtual environment. Thus, it makes a realistic experience.

On the other hand, it requires the instructor to provide feedback other than the effect of the decision on the patient. This limits learners to practice many cases at their convenient times of their choice. LiveBook provides valuable feedback on learner's performance automatically without the help of an expert. This allows the learner to practice more without consuming more time of the expert.

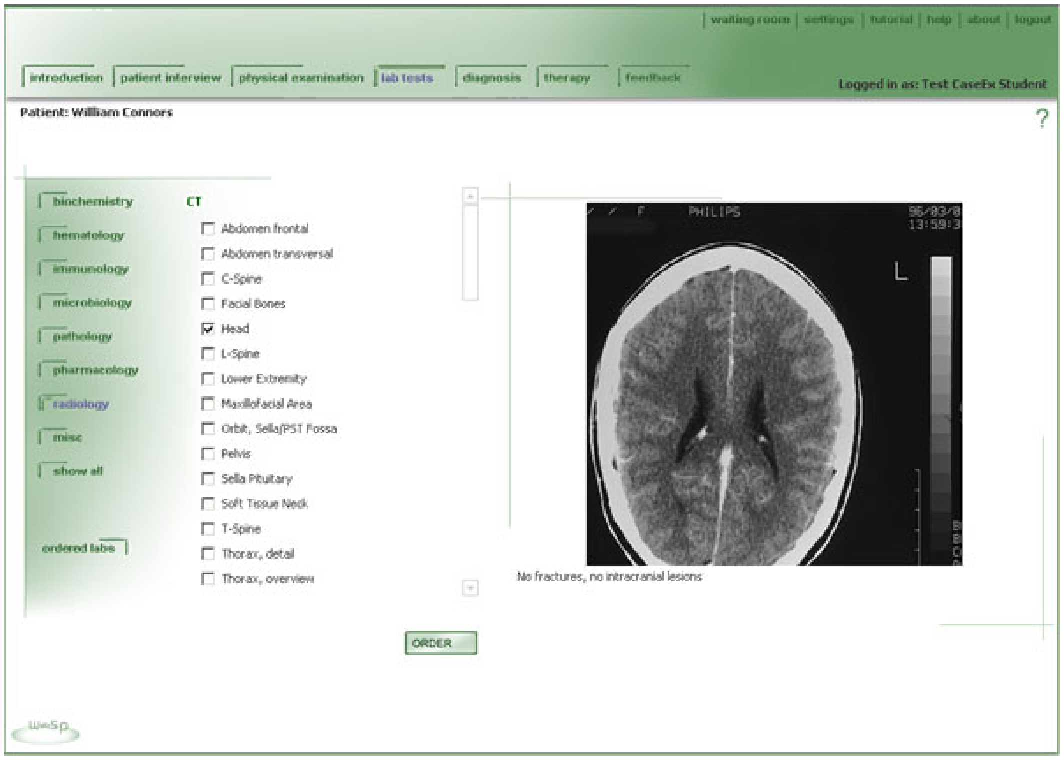


Figure A.6: A view of the Web-SP system

A.3.8 Web-SP

Web-SP is a web-based software developed for virtual patient simulation [33], [10]. The author can use a user-friendly interface in order to create a case. There are some template cases that the teachers can follow to create a case, they can also edit the existing cases to create new ones. Template cases are healthy patients that can be modified to generate virtual patients with a specific disease. According to their paper [33], it would take about 22 hours for an expert to develop a case. After a case is developed, the learners can select a case and study it. In each case, the learners can interview the patient or request physical exams or lab tests. In interview phase, the learner asks questions from the patient from a set of predefined questions. For the physical exams, the student selects the type of exam and then click on the appropriate body part, the results of exams are in a format of image, sound, text, or video. Laboratory tests can be ordered from a set of predefined tests. Lab results are shown as text, image, or video. After finishing the examination, the student is supposed to enter a diagnosis. Finally, the therapy screen comes up and the feedback screen is shown at the end. In the feedback section, the student gets feedback on the questions asked, requested tests, and the diagnosis. The Web-SP compares actions taken by the student with the recommended actions by the teachers. A log of all the actions undertaken by the learner is viewed at the end to be reviewed.

This system is created as a web application which does not require students to study at a specific location or time. It also puts the learner in the role of the doctor. Learners can also compare their own performance with an expert at the end of the study. Web-SP is also flexible enough to be used in different health-care disciplines like dentistry [32].

Although Web-SP simulates the role of the doctor, it does not enable the learners to ask any question they want. It will show and allow them to investigate a specific set of questions which is not the case in the real world. Web-SP also does not provide detailed automated feedback which could help the learners in their clinical reasoning. LiveBook scores each asked question based on how useful the question was in the process. It also provides feedback on the supporting evidence of the diagnosis including the missing evidence required to make that diagnosis.

A.3.9 OtoTrain

OtoTrain is is an otoscopy simulator for models of the auricle and ear canal [29]. It was designed in Unity software and has a real-time interaction with the models of the ear over the internet. The system can be accessed by an administrator, examiner, and trainee. The administrator role can manage the accounts and the program settings. The examiner can create exams by a collection of pathologies or uploading images from personal library. Then the trainees can access the system in two modes, study mode and exam mode. In study mode, they can browse different resources available in the system, and in the exam mode they can take an exam in a time period and the results will be directly sent to the examiner and the examiner can provide feedback through the system. The exams are multiple choice questions with media elements

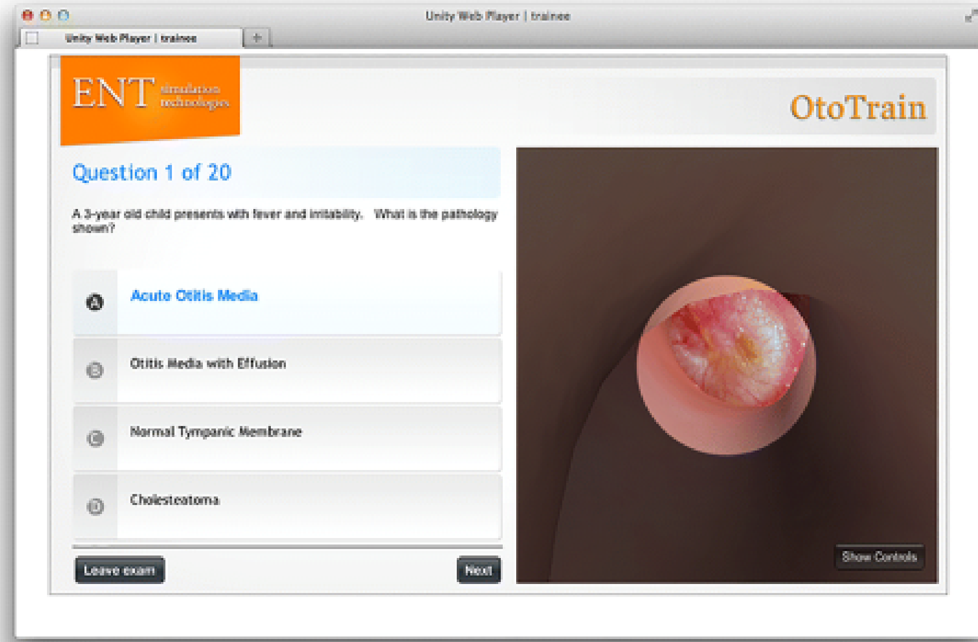


Figure A.7: A view of the OtoTrain system

as shown in Figure A.7 The trainees can also take a simulator tutorial where they will navigate a virtual otoscope with mouse and keyboard. The trainee must insert the speculum into the auditory canal and move it. The speculum simulates actual otoscope by limiting the view when it is moved around. Some feedback is automatically given to the trainee including time of completion, the percentage of tympanic membrane visualized, and the force against the ear canal.

This software is developed with Unity and is accessible by Unity Web Player which can be installed on browsers. This makes the software accessible through the internet. It simulates the process of otoscopy and provides feedback on how much pressure there was on the ear canal and how much of the tympanic membrane was investigated. This allows the learners to improve their performance in the real-world application by finding the best practices in the virtual world. It also provides images and videos in the exams which allow the learners to focus on the decision making when they have a good sight of the canal.

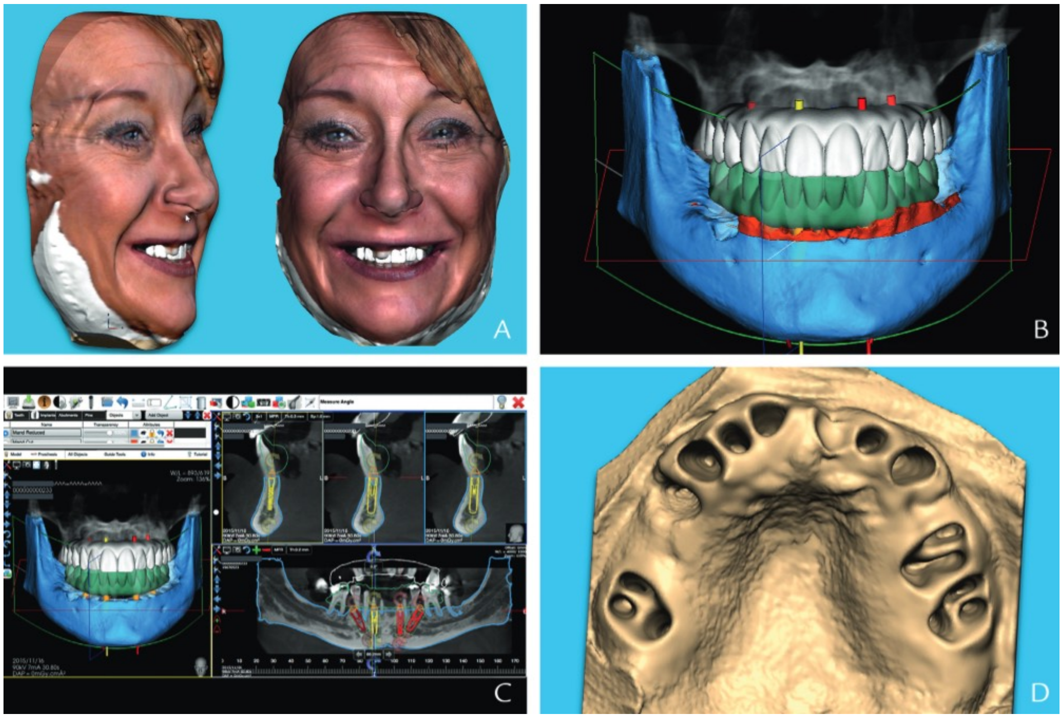


Figure A.8: A view of the 3D Virtual Dental Patient system

The system can be improved in handling the otoscope. In OtoTrain otoscope is move using keyboard and mouse which do not represent the full control over the device in the real world. This is important especially in the amount of force applied to the ear canal and when the doctor drives otoscope in the canal sensing the forces. During otoscopy talking and interacting with the patient is also a factor in diagnosis. The doctor can ask more question from the patient before he/she starts with the otoscopy. This interaction can continue during the process when the patient presents pain when pressure is applied to specific areas of the ear canal.

A.3.10 3D Virtual Dental Patient

Other Virtual Patient simulation systems try to resemble some part of the examination processes to help junior doctors developing their skills when students face the real world situations.

Authors in [11] developed a 3D virtual patient with 2D pictures and 3D extraoral facial scan. Using the pictures, this 3D virtual patient has craniofacial hard tissue and soft tissue with an exaggerated smile to make it more realistic. It is used to assist virtual diagnostic tooth arrangement process, and to gain patient's pre-treatment approval. It also helps as a surgical plan for computer-guided implant surgery. The system is using a low-cost extraoral 3D scanner to capture the facial soft tissue, instead of the more expensive extraoral 3D surface imaging system and merge its output with the 2D extraoral digital photograph with CAD software. The limitations of this system are first the consistency of all resources that are merged together in terms of accuracy. Also, the exaggerated smile is used to get information for diagnosing patient's maxillary lip position, hence, the head position and facial expression should remain the same during the 3D scanning.

This tool simulates the patient in both soft and hard tissue which allows for a realistic simulation. It can also be used to create a virtual treatment for the patient; thus, the learner can view the end result too.

The simulation only includes the end result of the surgery and it does not improve the decision making for a junior doctor. The system visualizes all the tissues when in reality the doctor does not have access to this information.

A.3.11 Interactive Simulation of Patient Cases (ISP)

The ISP system simulates an interactive patient [3]. It provides three main phases of history taking, physical examination and laboratory tests. In the history-taking phase, the user enters the question in the free-text format in natural language and small misspellings are acceptable by the system. Thousands of video clips are stored in the system to answer the questions asked. The virtual patient can interact with the user by showing frustration or being angry, or event leaving the physician's office if the user asks repeated/inappropriate questions or don't ask anything for a long time. The help option is also available to choose a question from a set of predefined questions. The student can also request a physical exam by choosing the exam and the body part they want the examination to be on. At the end, the proposed diagnosis is entered in free-text. If correct, the system provides short feedback on different diagnostic procedures. If wrong, the system generates feedback and the stu-

Figure A.9: A view of the ISP system

dent is motivated to continue with the case until reaches the correct diagnosis. Additional feedback is also produced for unnecessary, harmful, or the cost of requested lab exams or history questions asked by the learner.

The virtual patient cases are realistic and usable for medicine, dentistry and nursing students, as well as other areas. Different examinations are also available in the system. The feedback includes a detailed report on the learner's performance which can help the learner to optimize the diagnosis process with respect to the cost of the tests and patient's pain. The system is modular, so new features can be added easily which is an important factor in enabling the system to fit in different areas.

At the end of the cases, students were asked to fill out a printed survey that asks them about their thoughts on the system. It may not be convenient when the system is used by a big group of students. In addition, students are forced to enter questions in text format and they cannot speak to the patient. ISP also does not provide feedback on the clinical reasoning of the learner. The feedback includes the topics investigated in history taking, physical examinations performed and the ordered lab test and the total cost of the tests. But it does not include which questions could have been asked to help the diagnosis or how much helpful a question was in the process. This is while LiveBook provides the missing evidence required to make a final diagnosis in the report card.