People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
Chadli Bendjedid University
Faculty of Science and Technology
Computer science department

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة الشاذلي بن جديد
كلية العلوم و التكنولوجيا
قسم الاعلام الالي

جامعة الشاذلي بن جديد
UNIVERSITE CHADLI BENDJEDID

Master Thesis in Computer Science

Presented by

Khantouchi Ramzi

Specialty: Intelligent Computer Systems

THEME

Healthy Food Recommended System Based on Reinforcement learning

Defence on: 04 / 07 / 2021

Member of the jury:

Quantity	First And Last Name	Grade	University
President	Mrs. FERROUN ASSIA	MCB	Chadli Bendjedid El-Tarf
Supervisor	**Mrs. IBTISSEM GASMI**	**MCB**	**Chadli Bendjedid El-Tarf**
Examiner	Mr. CHEMAME CHAOUKI	MCB	Chadli Bendjedid El-Tarf

University Year: 2020/2021

Thanks

First of all we would like to thank Allah, the merciful who allowed us to carry out our end of study project.

As a testament to our gratitude and our deep respect, we then send our warm and sincere thanks to:

The members of the jury for their efforts and their care in our work.

Our supervisor, ***Mrs. IBTISSEM GASMI*** *for his availability, his advice and his encouragement which allowed us to carry out this work in the best conditions.*

Our friends who gave us their help in the elaboration of this thesis, we owe them a lot.

The teachers of our university and especially those of the IT department.

Our families for their support and encouragement over the years.

Dedication

I dedicate this thesis

To my father Said and my mother Khadija

To my sister Ibtissem who have always believed in me and never stopped supporting me.

To my brothers chamse edine and Islam and my sisters Roumaissa and Chaima

To my friends abdel kader and soufyane

To my colleagues abir , zineb , kawther, wahida , hounaida , nadjib and the whole class of Master2

To all those who are dear to me.

Table of content

Thanks I

Dedication II

Table of content III

List of figures IV

List of acronymes V

General Introduction 8

Chapter 1 : State of the art 9

I.1 Introduction 9

I.2 Reinforcement Learning 9

I.3 Learning and Planning 10

I.4 Markov Decision Process 10

I.5 Value Based Reinforcement Learning 11

I.6 Exploitation and Exploration 14

I.7 Reinforcement Learning in Recommendation Systems 14

I.8 Reinforcement Learning Applications in Video games 19

I.9 Conclusion 22

Chapter 2 :project study 23

1. Introduction 23

2. Problem Statement 23

3. API Functionalities 23

3.1 Main Functionalities 23

3.2 Subsidiary Functionalities 24

4. Proposed Framework 24

4.1 Problem Settings 24

4.2 DeepTaste Framework Architecture 25

4.2.2 Best Recommended Item Model 27

4.2.3 Top-N Recommendation Items Model 27
5. REST API Design 27
5.1 Recommendation Task Design 28
5.2 Use Case Diagram 29
5.3 Class Diagram 30
5.5 Sequence Diagrams 33
3. Conclusion 38
Chapter 3 : Realisation 39
1. Introduction 39
2. Development Environment and Tools 39
3. Technology that used 40
4. Application handling: 41
Admin graphic user interface 41
Client user interface 48
5. Conclusion 53
Conclusion and Perspectives 54
References 55

List of figures

Figure 1.1 : The agent–environment interaction in a Markov decision process……..13

Figure 2.1 : overview of the DeepTaste framework architecture …….……..29

Figure 2.2 : overview of the DeepTaste framework architecture……..……..30

Figure 2.3 : Overview of the REST API Architecture……...……..32

Figure 2.4 : the recommendation task design in the REST API……..33

Figure 2.5 : the use case diagram……..……..34

Figure 2.6 : class diagram……35

Figure 2.7 : Admin activity diagram……36

Figure 2.8 : Activity Diagram for Users……..37

Figure 2.9 : Sequence Diagram for the Agent training……..……..38

Figure 2.10 : Sequence Diagram for SignUp…….……..38

Figure 2.11 : Sequence Diagram for viewing appointments……..……..39

Figure 2.12 : Sequence Diagram for booking an appointment…….40

Figure 2.13 : Sequence Diagrams for Posting system……..41

Figure 2.14 : Sequence Diagrams for Posting system……..41

Figure 3.1: Admin login……...44

Figure 3.2 : Dashboard of admin account ……45

Figure 3.3 : Change Passwoard…….45

Figure 3. 1 acounts management…… 46

Figure 3. 2 search for accounts ……...46

Figure 3. 3 create account…… 47

Figure 3. 4 create account with permission graint……47

Figure 3. 5 save / edit/ delete option……48

Figure 3. 6 account details ……...48

Figure 3. 7 doctor account information_1…… 49

Figure 3. 8 doctor account information_2……49

Figure 3. 9 patient appointment……50

Figure 3. 10 add appointment ……50

Figure 3. 11 user history……..51

Figure 3. 12 log in request……...51

Figure 3. 13 profiel edit ……..52

Figure 3. 14 search for a doctor by name…….52

Figure 3. 15 get the appointments list…….53

Figure 3. 16 create an appointment …….53

Figure 3. 17 change an appointment date……54

Figure 3. 18 sign in page……..54

Figure 3. 19 home page ……...55

Figure 3. 20 profiel page……..55

List of acronymes

DQN : Deep Q Networ

kknn : K-Nearest Neighborhood

MDP: Markov Decision Process

REST: Representational State Transfer

API: Application Programming Interface

General Introduction

Due to the increasing of the World Wide Web and big data, recommender systems (RSs) are becoming more and more popular [1,2]. The purpose of these systems is to suggest different services to different users. In the existing literature, researchers have presented numerous recommendation approaches such as collaborative filtering (CF), content-based filtering (CBF) and hybrid techniques. Collaborative filtering is one of the most popular methods. It is based on the rating histories to calculate the similarities between users or items [3]. It suggests similar objects to those favored by the user in the past, or items which have been liked by similar users. The most popular technique in CF is the nearest neighbor, where a subset of *k* suitable users (or items) is chosen based on their similarities to the target one [4]. Various techniques consider the user—system interaction as a static process and only treat the current behaviors of the user, which negatively impact his experience over time. In order to address this issue we propose a Reinforcement learning framework called DeepTaste to recommend a personalized meal slates for users. The proposed method takes advantage of the nature of user—system interaction and formalize it as a Markov Decision Process. DeepTaste has Three Main components: a state encoder model which represent the user behaviors, the best recommended item model that predicts the next item that user might like according to his current state, and Top-N recommended items model which responsible for generating meal slate for the user. We also developed a REST API that provides several healthcare services such as booking appointments online with doctors. Our thesis will be decomposed into three chapters:

➢ In the first chapter, we introduce the theory of reinforcement learning paradigm and we present a detailed description of value based reinforcement learning, then, we talk about the success of reinforcement learning in the field of recommendation systems and other fields such as gaming.

➢ In chapter two, we describe in details the architecture of the proposed method as well as the REST API. The UML diagrams presenting the different views of the system are also given.

➢ In Chapter three, provides a detailed implementation of the proposed framework. It also discusses the results of the experiments.

➢ We finalize our thesis by a general conclusion and future works.

Chapter 1 : State of the art

I.1 Introduction

When we think about the nature of learning, The idea of learning by interacting with the environment may be the first one that comes to mind. When a baby plays, waves its arms, or looks about, it has no explicit teacher, but it does have a direct sensorimotor connection to its environment. Exercising this connection produces a wealth of information about cause and effect, about the consequences of actions, and about needs to be done to achieve goals. In our lifetime, these interactions are undoubtedly kinds of important sources of knowledge about our environment and ourselves. Whether we are learning to drive a car or having a conversation, we are acutely aware of how our environment responds to our actions and we try to influence what happens through our actions. Learning from interaction is the basic idea of almost all learning and intelligence theory [5].

I.2 Reinforcement Learning

Reinforcement learning is learning what to do how to map situations to actions so as to maximize a numerical reward signal. The learner is not told what actions to take, but must be tested to discover which actions will produce the greatest reward. In the most interesting and challenging cases, actions may not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics trial-and-error search and delayed reward are the two most important distinguishing features of reinforcement learning[5]. Reinforcement learning is different from supervised learning. Supervised learning is the type of learning studied in most current research in the field of machine learning. Supervised learning is learning from a training set of labeled examples provided by a knowledgeable external supervisor. Each example is a description of the situation and a specification (label) of the correct actions the system should take in response to the situation. This is usually used to identify the category to which the situation belongs. The purpose of this type of learning is to allow the system to infer or generalize its response in order to act correctly in situations that do not exist in the training set. This is an important type of learning, but it is not suitable for learning from interaction. In interactive problems, it is often impractical to obtain examples of the desired behavior that are correct and represent all the situations in which the agent must act. In uncharted

territory (where learning is expected to be most advantageous), agents should be able to learn from their own experience.

Reinforcement learning is also different from what machine learning researchers call unsupervised learning, which generally looks for hidden structures in unlabeled data sets. The terms supervised learning and unsupervised learning seem to categorize machine learning paradigms in detail, but this is not the case. Although people may tend to think of reinforcement learning as a form of unsupervised learning because it is not based on examples of correct behavior, reinforcement learning is trying to maximize reward signals instead of trying to find hidden structures[5].Being able to unravel the structure from the agent's experience certainly helps reinforcement learning, but does not in itself solve the problem of reinforcement learning maximizing reward signals. Therefore, we believe that reinforcement learning is the third paradigm of machine learning, as well as supervised learning and unsupervised learning and other paradigms.

I.3 Learning and Planning

Various tasks in artificial intelligence and control can be formalized as sequential decision processes.We refer to the decision-making entity as the agent, and everything except the agent as its environment.At each time-step t the agent receives observations *st* S from its ∈ S from its environment, and executes an environment, and executes an action *at* ∈ S from its environment, and executes an A according to its policy. The environment then provides a feedback signal in the form of a reward *rt*+1 ∈ S from its environment, and executes an R. This time series of actions, observations and rewards defines the agent's experience. The goal of reinforcement learning is to increase the future reward of the agent based on past experience.

I.4 Markov Decision Process

Markov Decision Processes are a classical formalization of sequential decision making, in which actions affect not only the immediate reward, but also subsequent situations, or states, and through those future rewards. Thus Markov Decision Processes involve delayed reward and the need to trade off immediate and delayed reward. Markov Decision Processes are a mathematically idealized form of the reinforcement learning problem , which can be precisely expressed in theory. We can define a Markov Decision Process as a tuple $\langle S, A, P, R, \gamma \rangle$ where :

➢ S is a finite set of states

➢ A is a finite set of actions

➢ P is state transition probability matrix, $Pss' a = P[St+1=s' | St=s , At=a]$ (1.1)2

➢ R is a reward function, $Rs a=E[Rt+1|St=s , At=a]$ (1.2)

➢ γ is discount factor γ∈[0,1]

Markov Decision Processes aim to directly construct problems that learn from interaction to achieve goals. The learner and decision maker is called the agent. The things it interacts with, including all things other than the agent, are called the environment[1]. These continuous interactions, the agent chooses actions, and the environment responds to these actions and presents new situations to the agent. The environment also generates rewards, that is, the agent seeks to maximize over time through its chosen actions.

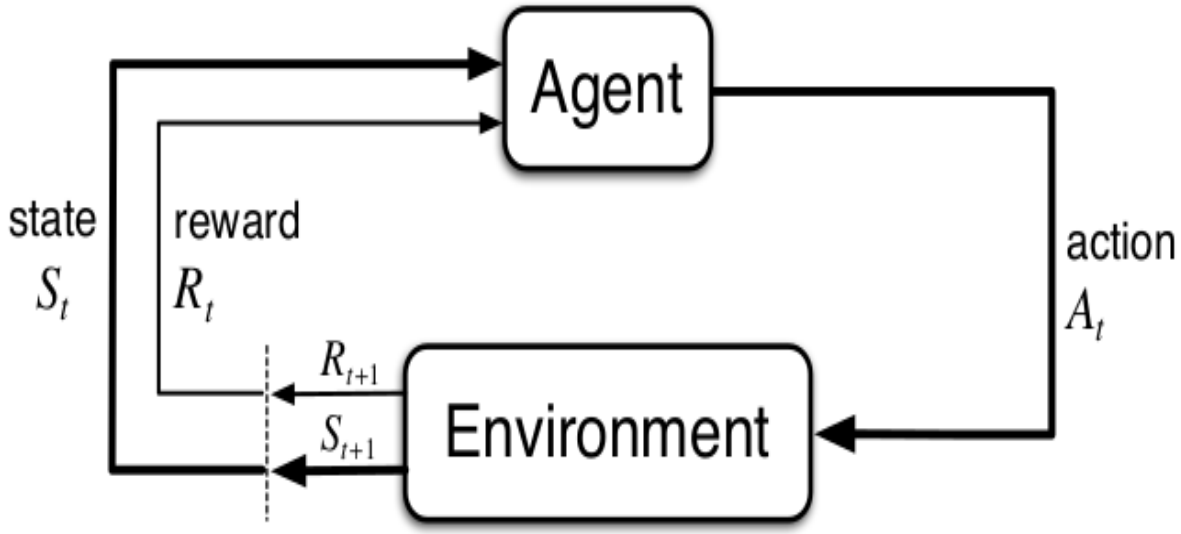

Figure 1.1 *The agent–environment interaction in a Markov decision process.*

I.5 Value Based Reinforcement Learning

The most successful example of reinforcement learning is the use of value functions to summarize the long-term consequences of specific decision-making policies. The value function $V\pi(s)$ is the expected return from state s when following policy π. The action value function $Q\pi(s , a)$ is the expected return after selecting action a in state s and then following policy π,[2]

$V\pi(s)=E\pi[Rt \forall st=s]$ (1.3)

$Q\pi(s , a)=E\pi[Rt|st=s , at=a]$. (1.4)

The Value-based reinforcement learning algorithms interactively update an estimate V (s) or Q(s, a) of the value function for the agent's current policy. The updated value function can be used to improve the policy, for example by greedily selecting actions for the new value function. This cyclic process of policy evaluation and policy improvement is at the heart of all value-based

reinforcement learning methods. In model-free reinforcement learning algorithms such as Monte-Carlo evaluation and
temporal-difference learning, the value function is updated through sample backups. At each time step, a single action sample is obtained from the agent policy, and a single state transition and reward is sampled from the environment[2].

I.5.1 Monte-Carlo Evaluation

Monte-Carlo evaluation provides a particularly simple, model-free method for policy evaluation. The value function of each state s is estimated by the average return over all episodes that visited the state s. At each time-step, Monte-Carlo updates the value of the current state to the return [6]. However, this return depends on the specific action and the specific state transition sampled in each subsequent state, which can be a very noisy signal. Monte-Carlo gives an unbiased, but high variance estimate of the real value function.

I.5.2 Temporal Difference Learning

Temporal-difference learning is a model-free method for policy evaluation. It bootstraps the value function from subsequent estimates of the value function. Bootstrapping method is used for reduce the variance of an estimate. In the TD(0) algorithm, the value function is bootstrapped from the next time step. Instead of waiting until full return is observed, the next state value function is used to approximate expected return. TD_error $\delta t = rt + 1 + V(st+1) - V(st)$ is measured between the value in state st and the value in subsequent state $st+1$, plus any rewards accumulated on the way $rt+1$.

$$\delta t = rt + 1 + V(st+1) - V(st) \quad (1.3)$$

$$\Delta V(st) = \alpha \delta t \quad (1.4)$$

where α is the learning rate.

I.5.3 TD(λ)

The concept of the TD (λ) algorithm is to bootstrap state values) algorithm is to bootstrap state values with subsequent values at various stages in the future. The parameter λ) algorithm is to bootstrap state values determines the range of time during which bootstrap occurs. At one extreme, TD (0) bootstraps the value of the state only in the direct successor state. The other extreme TD (1) updates the status value with the final return. Same as Monte Carlo evaluation. Eligibility tracking z (s) is maintained in each state to implement TD (λ) algorithm is to bootstrap state values) in stages. Qualification tracking represents the total credit that must be primarily allocated for subsequent errors in the valuation. Combine the latest heuristics with frequency heuristics. The most frequently visited and most recently accessed condition will be

best qualified. Each time the state is accessed, the eligibility trace will increase and the constant parameter λ) algorithm is to bootstrap state values will decrease at each time step. Every time you see a difference between the predicted value and the subsequent value, TD_error δt is generated. Value functions for all states are proportionally updated based on TD_error and state eligibility.

$zt(s)=\lambda zzt\ -1(s\ )if\ s\neq st\vee\lambda zzt-1(s)+1\ if\ s=st$ (1.5)

$\delta t=rt+\ 1+Vt\ (st+1)-Vt\ (st)$ (1.6)

$\Delta Vt\ (s)=\alpha\delta t\ zt(s).$ (1.7)

I.5.4 Sarsa(λ)

The Sarsa algorithm combines policy improvement with temporal-difference learning . The TD (λ) algorithm is to bootstrap state values) algorithm updates an action value function and the last action value is used to select an action. The greedy policy is used to combine exploration (a random action with a selection probability) and utilization (argmax Q (s, a) with a selection probability of 1). The action-value function is updated from each experience tuple (s, a, r, s', a'), using the update rule TD (λ) algorithm is to bootstrap state values) of the value of the state action. If all states are visited indefinitely and an appropriate step size is selected, the Sarsa algorithm converges to the optimal policy.

I.5.5 Value Function Approximation

In a large environment such as our messy world, it is impossible or impractical to know the value of each state s. In this case, it is necessary to represent the state more compactly by using some set of feature for the state s. Then, the value function can be approximated by the features vector X of state s and the parameter θ. A good practice to approximate the value function is to use a linear combination between features vector and parameters.

$V\ (s)=X\ .\theta$ (1.8) Non-linear functions such as neural networks can be used to approximate the value function using the features vector and weights of the neural network. When it comes to approximate the value function using features vector, errors could be attributed to any or all of those features. Gradient descent can be used to this problem to update the parameters in order to minimize the mean-squared evaluation error . Monte-Carlo evaluation with linear function approximation provides a particularly simple case. The parameters are updated by stochastic gradient descent, with a step-size of α[2],

$\Delta\theta=-\alpha$

$2\ \nabla\theta(Rt-V\ (st\ ))2$ (1.9)

I.6 Exploitation and Exploration

In reinforcement learning, Exploitation means that when an agent take an action a in a specific state s according to its policy in order to maximize rewards while exploration means that the agent chose to explore to find more information about the environment. Balancing between exploitation and exploration is a challenging problem in reinforcement learning.

I.6.1 Epsilon-greedy

Epsilon-greedy is a simple method to balance between exploitation and exploration by choosing one of them randomly. Epsilon refers to the probability of chose to explore, exploits most of the time with a small chance of exploring.

I.6.2 Upper-Confidence-Bound

It is necessary to explore, because the accuracy of the estimate of the share value is always uncertain. Greedy actions are currently the best actions but some of the other actions at may be better. "The selection of greedy actions forces the attempt of non-greedy actions, but indiscriminately, and does not prefer those that are close to greedy or particularly uncertain actions. how close your estimates are to the maximum and the uncertainty in these estimates. An effective way to balance between exploitation and exploration is to use the following formula,

$At=argmax\ a\ [Qt\ (a)+c\sqrt{}\ \ln(t)\ /Nt\ (a)\ ]$ (1.9)

where $Nt(a)$ is the number of times that action a has been selected prior to time t and c >0 is the control factor of the exploration, the square root term is to measure the uncertainty in the estimate of the action value of a[1].

I.7 Reinforcement Learning in Recommendation Systems

I.7.1 Recommender Systems

In our daily life, it is not very uncommon for us to be faced with situations in which we have to make decisions without a prior information about our choices. In that case, it seems very necessary to rely on the recommendations of others with experience [9]. This was the rationale behind the first RS, Tapestry [10], and they termed it as collaborative filtering *(CF)*. Later, the term was extended to recommender systems to reflect two facts [9]:

•the method may not collaborate with the user

- the method may suggest interesting elements instead of filtering them.

By definition, recommender systems are software tools and algorithms that suggest items that might be of interest to the users [11]. Another important approach toward the recommendation problem is content based filtering (CBF). In content based filtering, the idea is to recommend items similar to the user profile, which is a structured representation of user interests [12, 13]. Due to the problems of the two methods (i.e., CF and CBF), specifically cold-start (when user/item is new) and serendipity (having a diverse range of recommendations), hybrid methods was proposed to cover these problems [11]. In general, we call these methods traditional RS because they are unlikely to be able to deal with current recommendations due to their sharp problems, namely cold start, contingency, scalability, low quality and static recommendations, and high computational overhead. Has a large number of users and items.

I.7.2 Reinforcement Learning in Recommendation

The essence of user interaction with recommender system is sequence [14]. The problem of recommending the best items to users is not only a prediction problem, but also a sequential decision-making problem [15] This shows that the recommendation problem can be modeled as Markov Decision Process and solved using reinforcement learning methods. As mentioned earlier, in a typical reinforcement learning environment, the agent aims to maximize digital rewards through interaction with the environment. This is analogous to the recommendation problem where the recomender system algorithm tries to recommend the best items to the user and to maximize the user's satisfaction. Therefore, the recommender system algorithm can play the role of the reinforcement learning agent and everything outside this agent, including the users of the system and items, can be considered as the environment for this agent. It is almost infeasible to apply traditional tabular reinforcement learning algorithms to today's recommender systems with huge action and state spaces. Instead, with the development of Deep Reinforcement Learning algorithms, it is becoming an emerging trend among the recommender system community to employ reinforcement learning techniques [16].

I.7.2.1 Temporal Difference Methods

Q-learning has been a popular reinforcement learning algorithm among the recommender system community [17]. WebWatcher [17] may be the first recommendation system algorithm to use reinforcement learning to improve the quality of recommendations. They simply model the web page recommendation problem as an reinforcement learning problem and use Q-learning to improve the accuracy of their basic Web RS, which uses a similarity function (based on TF-IDF) to recommend similar pages to users. A decade later, Taghipour and Kardan [18] extend

this idea to recommend personalized web pages to the users. More precisely, to tackle the state dimensionality problem, they borrow the N-gram model from the web usage mining literature [19] And use a sliding window to indicate status. Later in [20], they improved their work by incorporating conceptual information into a use-based recommender system network. Another TD method is [21], in which a travel agency was developed to recommend personalized travel to tourists. The method consists of two main modules:

personalized students, responsible for learning static and dynamic information from users, personalized rankings and responsible for generating recommendations using Q-learning. Although the work is one of the first attempts to conduct small-scale online experiments with real users, it is unclear how they deal with recommendation problems with large states and action spaces. In addition, some technical details related to use reinforcement learning, including the reward function, are not yet clear. Reinforcement learning based recommender system session was proposed in [22]. The main difference here is that Q-learning is used instead of repeated policy algorithms to optimize politics. They maintain manageable and action spaces when limiting them to defined numbers. The main contribution of RLWREC [23] is to present a state compression model to address the dimensional problem of state space. In particular, the idea is to group the songs depending on the performance of similar users, and then replacing the songs with the song groups in the learning stage. The Kumean algorithm is used to group the songs.

SARSA is another TD algorithm used by some recommender systems [24, 25, 26]. The recommender system network in [24] has two main units: global and local. The global unit is responsible for understanding global system trends such as the most popular products, while the local unit tracks each customer individually. The system uses a weighted method to combine the local model and the global model to select the next page for recommendation. An obvious issue with this job is scalability, because it's unclear how they want to track all users globally. SARSA (λ) is a rough solution version of the original SARSA algorithm, and was used in [60] to develop a Web recommender system based on a custom ontology. The system uses a weighted method to combine the local model and the global model to select the next page for recommendation. An obvious issue with this job is scalability, because it's unclear how they want to track all users globally. SARSA (λ) is a rough solution version of the original SARSA algorithm, and was used in [25] to develop a Web recommender system based on a custom ontology. The purpose of RS is to provide students with a learning path that suits their specific requirements and characteristics. The purpose of RS is to provide students with a learning path that suits their specific requirements and characteristics. They use the N-gram model to solve the problem of state dimension like [18].

There are also works that test Q-learning and SARSA at the same time to optimize their strategies [27]. For example, emotion-based playlist generation is described as an reinforcement learning problem in [27]. To manage the state space, similar to [18], the N-gram (sliding window) model is used to model states, that is, each state contains information about the emotional category of the last K songs of the user.

I.7.2.2 Dynamic Programming Methods

DP is another tubular method that has been utilized in [28, 15, 29, 30]. Among the early works that formulates the recommendation problem as an Markov Decision Process is [28]. In fact, this task illustrates the benefits that can be obtained using Markov Decision Process for recommended problems with an example that guides the user from the airport. One of the earliest valuable attempts to model a similarly recommended problem into Markov Decision Process is [15] Since the model parameters of the Markov Decision Process based recommenders are unknown and the cost of implementing them in a real environment to learn them is very high, they recommend using a predictive model that can provide initial parameters for Markov Decision Process. This predictive model is a Markov chain in which states and transition functions are modeled based on observations in the data set. The basic version of this Markov chain uses the maximum probability to estimate the transfer function, but they believe that it faces the problem of data scarcity. Therefore, using three techniques, jumping, grouping and hybrid modeling, the basic version is improved. Then use the predictive model to initialize the Markov Decision Process based recommender. To address the dimensional problem, the last K item is used to encode state information. They tested their way of performance using online research [31].

I.7.2.3 Monte Carlo Methods

Monte Carlo is the latest tabular method and has been used in some recommender systems based reinforcement learning [32, 33, 34]. DJMC [32] is an reinforcement-based music playlist recommender. In order to solve the problem of dimensionality, each song is modeled as a vector of song descriptors (spectral auditory), which includes the song's spectral fingerprints, rhythm characteristics, overall volume, and weather information that changes over time. In addition, in order to speed up the learning process, reward functions are also considered, such as the listener's preference for individual songs and their song transition patterns. The DJMC architecture consists of two main parts: learning the parameters of the listener (their preferences for songs and transitions) and planning the song sequence [31]. The unit of learning is divided into two parts: initialization and flight learning. During the initialization phase, the listener will

be asked about the settings of their songs and transitions. After initialization, the learning process starts with playing the song for the listener and requesting their feedback on the song. The planning step is responsible for selecting the best song to be recommended. For this, Monte Carlo Tree Search is used. If the song space is too large or the search time is limited, group the songs (using k-means). A similar work to DJMC is PHRR [33], they use a mixture of Weighted Matrix Factorization (WMF) [35] and Convolutional Neural Network (CNN) to extract song features. The goal of DivFMCTS [30] is to propose a method for optimizing various topN recommendation problems. This method constitutes the configuration of two cyclic stages [31]. The first heuristic seeks space for items to find the best main recommendations using the Monte Carlo Tree Search algorithm. Then we generalize these findings by neural networks. Two ways to solve scalability problems when searching for all items, use two ways to disassemble the structural pruning and problems. In addition, a Deep Learning model Gated Recurrent Units (GRU) is used to encode user preference information in a state.

I.7.2.4 Compound Methods

In a rare but interesting application, [36] Use reinforcement learning to recommend learning activities in smart classrooms. In particular, a cyber-physical social system was established to monitor students' learning status by collecting students' multi-modal data, such as test scores, heartbeat, and facial expressions, and then recommend appropriate learning activities for them [31].

SlateQ [37], is a recommendation system based on reinforcement learning. In order to decompose the Q_value of the list into a combination of item Q_values, they assume that:

- The user only consumes one item in the list.
- The reward depends only on the item consumed. Using this decomposition, they showed that TD methods, such as SARSA and Q-learning, can be used to maximize long-term user engagement.

They also propose a flexible environment simulator, called RecSim, which simulates the dynamics of both users and RSs. In an interesting RS application that is based on SlateQ [31]. In [38], they design an RL-like framework for an activity recommender for social-emotional learning of students. More precisely, the recommender agent recommends a series of activities from the activity database. The instructor then selects an activity from the list and the group completes it. In response, the group provides feedback on the activity performed and the agent updates its policy accordingly [31].

In [39], a task-oriented dialogue management system was proposed and applied to different recommendation tasks. Two methods of dialogue management are proposed: segmentation-based and state-based. In the former, the user base is segmented based on context, such as demographics and purchase history, and each segment has a separate strategy. The last method relies on concatenating the agent's beliefs about conversation history, user intent, and context. Therefore, this belief vector provides a single strategy for all users. This work was tested on reference [40] in the field, which included various recommendation tasks, such as recommending restaurants in Cambridge or San Francisco [41]. The authors of EDRR [41] describe three elements common to all reinforcement learning approaches: embedded state representations and policies. They are arguing that it is impossible to directly train an embedded module using the other 2 modules because the reinforcement learning method has a large slope of variance.

I.7.2.5 Q Fitted Methods

There are also some reinforcement learning based algorithms [43] that use approximate methods (adjusted Q) for strategy optimization. In clinical applications [43], reinforcement learning is used to recommend treatment options for lung cancer patients, with the goal of achieving the greatest patient survival rate. They treat the treatment of patients with advanced non-small cell lung cancer (NSCLC) as a two-line treatment, in which the task of the reinforcement learning agent is to recommend the best treatment plan in each line of treatment, and the best timeline therapy to initiate the second line of treatment. For agent reinforcement , support vector regression (SVR) is used to fit the Q function. Since the original SVR cannot be applied to censored data, they modified the SVR using the loss function.

I.8 Reinforcement Learning Applications in Video games

The application of reinforcement learning is still far from conventional and generally requires both art and science. Making applications simpler and more direct is one of the goals of current reinforcement learning research. On the other hand on gaming reinforcement learning is widely used. A lot of companies implemented reinforcement learning algorithms on gaming sector and they achieved and fascinating results. Those algorithms were able to surpass the human level professionals in various complicated games.

I.8.1 Atari Games

A team at Deep mind created an agent called DQN and later on thy used it to show how a reinforcement learning agent can achieve a high level of performance on any of a collection of different problems without having to use different problem-specific feature sets. To demonstrate this, they let DQN learn to play 49 different Atari 2600 video games by interacting with a game emulator [5]. DQN learned different policies for each 49 games because the artificial neural network weights were reset to random values before learning on each game, the same raw input network architecture and parameter values. DQN has surpassed the human level in most of these games. The games are similar in that they watch and play streams of video images, but they differ significantly. Their behavior has different effects, and the dynamics of different state changes required different policies to achieve high scores. Deep convolutional neural network have been aesthetically pleasing to transform raw inputs common to all games into specialized functions to represent the action values needed to play at the high level of DQN achieved in most games [5].

I.8.2 Mastering the Game of Go

The ancient Chinese game of Go has challenged artificial intelligence researchers for many decades. Methods that achieve human-level skill, or even superhuman-level skill, in other games have been failed to achieve human-level in the game of Go. However, until recently, there was no Go program close to the human Go master level. A team of researchers at DeepMind [7] combine deep neural networks , supervised learning, Monte Carlo tree search, and reinforcement learning to develop a program called AlphaGo. AlphaGo had defeated the European Go champion Fan Hui 5 games to 0. This was the first win of a Go program for professional Go players with no handicap in a full Go game. Later on, a similar version of AlphaGo overtook 18-time world champion Lee Sedol to win four out of five in a challenge match. Artificial intelligence researchers thought that it would be many more years, perhaps decades, before a program reached this level of play. Later on the same team developed a program called AlphaGo Zero [8] which use only reinforcement learning and no human data or guidance beyond the basic rules of the game. AlphaGo Zero was able to defeat the previous go program AlphaGo.

I.8.3 Alpha Zero

In 2017, a team of researchers at deepMind developed a successor of AlphaGo Zero which is based on tabula-rasa reinforcement learning and self play. The researchers generalize this approach into a single AlphaZero algorithm that can achieve, tabula rasa, superhuman

performance in many challenging domains. Starting from random play, and given no domain knowledge except the game rules, AlphaZero achieved within 24 hours a superhuman level of play in the games of chess and shogi as well as Go, and convincingly defeated a world-champion program in each case [42].

I.8.4 Hide and Seek

OpenAI conducted experiments were targeted to train a series of reinforcement learning agents in mastering the game of hide and seek. In the target setting**,** the task of the agents is to participate in a two-team hide-and-seek game in a physics-based environment. The task of the hiders is to avoid the sight of the seekers, and the task of the seekers is to keep the sight of the hiders. There are some objects scattered in the environment, and the agents can grab them and lock them in place.There are objects scattered throughout the environment that the agents can grab and also lock in place. There are also randomly generated immovable rooms and walls that the agents must learn to navigate. The OpenAI environment contains no explicit incentives for agents to interact with objects. Agents are given a team-based reward; hiders are given a reward of +1 if all hiders are hidden and -1 if any hider is seen by a seeker. Seekers are given the opposite reward, -1 if all hiders are hidden and +1 otherwise. To confine agent behavior to a reasonable space, agents are penalized if they go too far outside the play area. During the preparation phase, all agents are given zero reward. To train the hide and see agents, OpenAI researchers leveraged the training infrastructure that was used in other multi-player games like OpenAI Five and Dacty. This type of infrastructure relies on a policy network in which agents are trained using self-play, which acts as a natural curriculum as agents always play opponents of an appropriate level. Agent policies are composed of two separate networks with different parameters a policy network which produces an action distribution and a critic network which predicts the discounted future returns. Each object is embedded and then passed through a masked residual self attention block, similar to those used in transformers, where the attention is over objects instead of over time. Objects that are not in line-of-sight and in front of the agent are masked out such that the agent has no information of them. As AI agents compete against each other in the environment explained before, they didn't only master hide and seek but they developed as many as six distinct strategies that were not part of the initial incentives.

Initially, hiders and seekers learn to crudely run away and chase. After approximately 25 million episodes of hide-and-seek, the hiders learn to use the tools at their disposal and intentionally modify their environment. They begin to construct secure shelters in which to hide by moving many boxes together or against walls and locking them in place. After another 75

million episodes, the seekers also learn rudimentary tool use; they learn to move and use ramps to jump over obstacles, allowing them to enter the hiders' shelter. 10 million episodes later, the hiders learn to defend against this strategy; the hiders learn to bring the ramps to the edge of the play area and lock them in place, seemingly removing the only tool the seekers have at their disposal. Similarly, After 380 million total episodes of training, the seekers learn to bring a box to the edge of the play area where the hiders have locked the ramps. The seekers then jump on top of the box and surf it to the hiders' shelter; this is possible because the environment allows agents to move together with the box regardless of whether they are on the ground or not. In response, the hiders learn to lock all of the boxes in place before building their shelter. The fascinating thing about the new behaviors developed by the Hide and Seek agents is that they have evolved completely organically as part of an automated curriculum guided by internal competition. The execution of actions that occur in almost all cases was better than that learned by implicit motives.

I.9 Conclusion

Reinforcement learning is an interesting machine learning approach to understanding and automating

goal-directed learning and decision making. It does not require exemplary supervision or a complete model of the environment, but is distinguished from other approaches by focusing on learning by trial and error which means that the agent learns from the direct interaction with the environment. Reinforcement learning use Markov Decision Process framework to model the interaction between the learning agent and its environment in terms of states, actions and rewards. Markov Decision Process framework aims to be a simple way of representing the nature of the artificial intelligence problems. Reinforcement learning paradigm describes the basic idea that underlies almost every theory of learning and intelligence. Combining Reinforcement learning with deep learning has enabled reinforcement learning to be applied in a large scale problems and crack many challenging problems for artificial intelligence such as the game of Go. Reinforcement learning approach is a key ingredient to realize the dream of Artificial General Intelligence (AGI).

Chapter 2 :project study

1. Introduction

In this chapter, we will start by defining the main functionalities of our REST API that we developed, next we will give a detailed architecture of the Recommender agent and later on we will give a detailed design for each functionality in the REST API and we will finish the chapter by a conclusion.

2. Problem Statement

Due to the increasing of online services such as online social networks, news, shopping, booking, E-learning and healthcare services, it poses a significant challenge for users to find the items that match their interests. Recommender systems become vital organ for giant companies such as google, Facebook, Tencent, Amazon and Netflix. Since that recommender systems have an important role for such companies and the life of users, its critical to build a robust recommender systems that satisfy users interests and match their needs. Recommendation approaches such as Collaborative Filtering take the user – system interaction as a static process and address user current behaviors which damage the user experience in the long run. In order to address this issue we propose a Reinforcement learning framework called DeepTaste to recommend a personalized meal slates for users. The proposed framework takes advantage of the nature of user—system interaction and formalize it as a Markov Decision Process. DeepTaste has Three Main components: a state encoder model which represent the user behaviors, the best recommended item model that predicts the next item that user might like according to his current state, and Top-N recommended items model which responsible for generating meal slate for the user.

3. API Functionalities

In this section we are going to address all the main and subsidiary functionalities in our REST API which provides services in the field of healthcare.

3.1 Main Functionalities

- Admin dashboard

- Food Recommender Agent DeepTaste
- Rate Food
- User Profile management (create, edit, add health status)
- Appointments management (book,view, cancel)
- Search for doctors
- Rate Doctors
- Posts management (view,create,edit,delete)

3.2 Subsidiary Functionalities

- Performance
- Security (Password Encryption)
- Compatibility with any platform (Linux,Windows and Mac OS)
- Easily integrated at any type of applications (Desktop apps, web apps, mobile apps,..)
- Easy to add new Functionalities

4. Proposed Framework

4.1 Problem Settings

We formalize the recommendation task as Markov Decision Process (MDP). We defined the state, action, and reward as follows:

- State $s_t \in S$represents the observed user positive interaction history (liked food history of user).
- Action $a_t \in A$represents the recommended item that the agent select according to current user state (user positive interaction history).
- Reward $r_t \in R$is the utility function that we want to maximize which defined as the user feedback. It takes +1 for each item liked by the user, if the liked item is the item selected

by the agent, a high reward score is given on the other hand it takes -1 for each disliked item as well as a high penalty score if the disliked item is the one that selected by the agent.

The figure below describes an overview of the DeepTaste framework architecture.

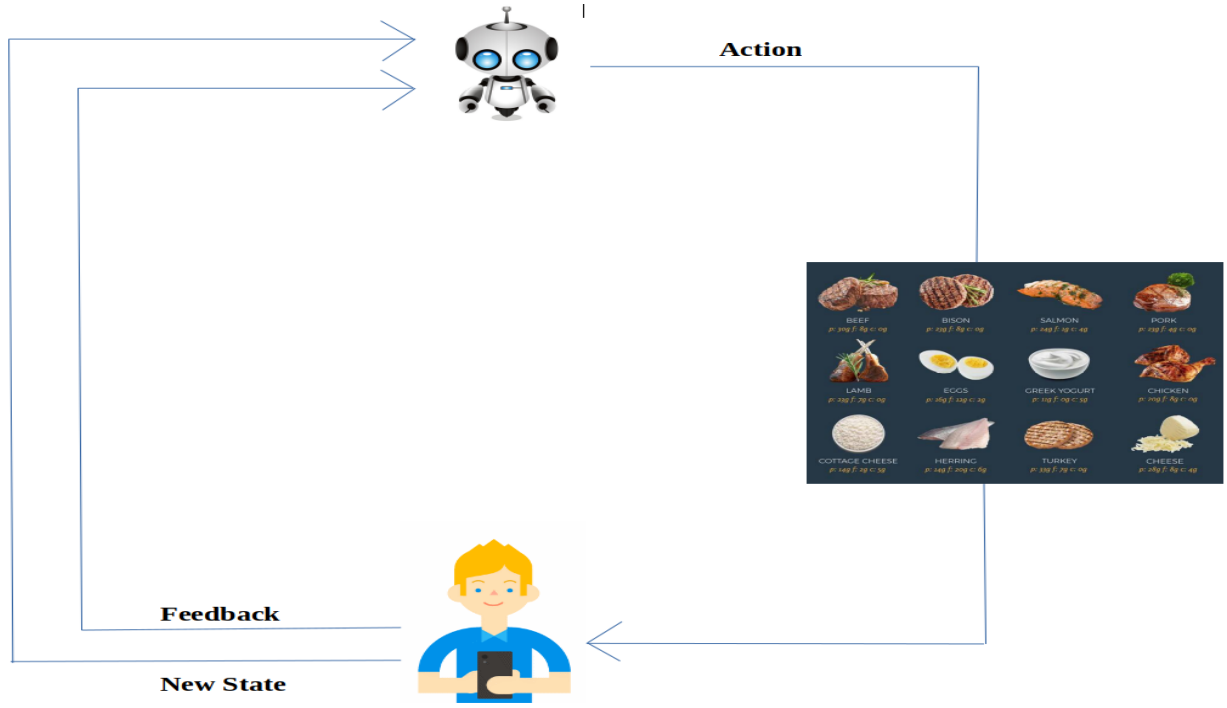

Figure 2.1 : overview of the DeepTaste framework architecture.

4.2 DeepTaste Framework Architecture

In this section we are going to present the technical details of each component in the proposed framework.

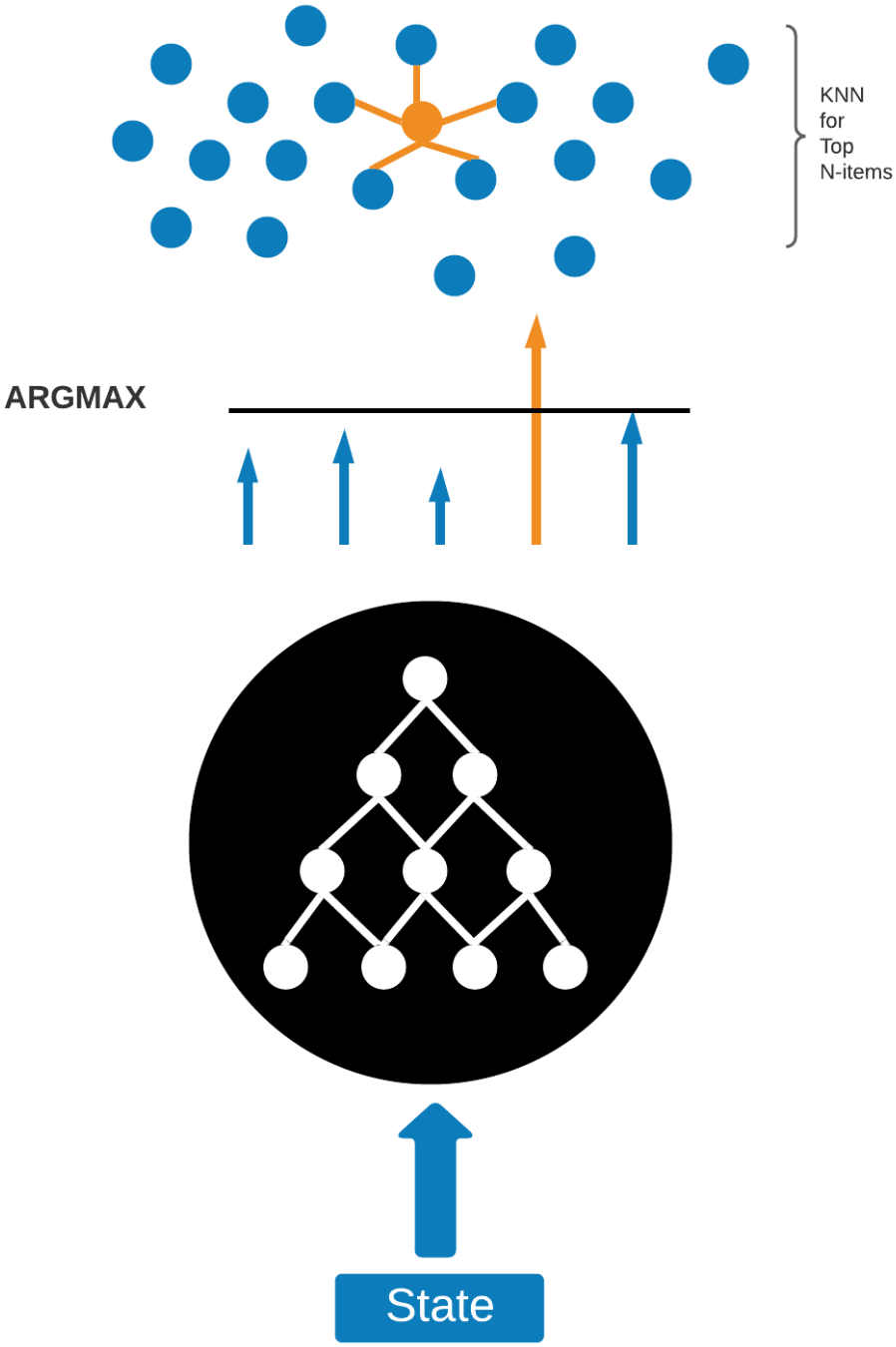

Figure 2.2: Detailed framework architecture

4.2.1 State Encoder Model

In reinforcement learning, a good state representation is very important part ant it affect the learning task because the agent makes decisions based on the current state. In our framework we defined the state as the latest 10 user positive interactions (liked foods) history. We take the ingredients information of the latest 10 items (food) that user liked, then we use **Doc2Vec** [44] encode each one of them into $R^{1\times10}$vector which represents item feature vector, after that we

take the feature vectors of the tenth items and concatenate them into one$R^{1\times100}$features vector which represent user current state.

Doc2vec is a technique that heavily based on **word2vec** [44], to create a numerical representation of a document regardless of its length. Its very nice technique and easy to use and gives good results.

4.2.2 Best Recommended Item Model

For the recommendation task we use **DQN** algorithm which takes user state features vector as an input and output a Q_value for each item (food) and contains one hidden layer. The DQN model tries to learn action-value function Q(s) in order to predicts action-value Q for each action (item) according to the user state, and then we use argmax(Q_values) to select the best candidate to recommend to the user. We also use Epsilon-greedy algorithm to balance between exploration and exploitation. The algorithm learns the action-value function online through the interaction with users.

4.2.3 Top-N Recommendation Items Model

Since we want to recommend top N items(food) for the user, we take features vector of the best candidate item that recommended by DQN agent in the previous step, and we use **K-Nearest Neighbors K-NN** to find the closest k actions (items) to the best candidate, then we recommend them with the best candidate item as the top N recommended items(food) in our case we will recommend 10 items for the user.

5. REST API Design

In this section we are going to give a detailed design for all the functionalities in our Rest API as well as how the recommender system is integrated in the API.

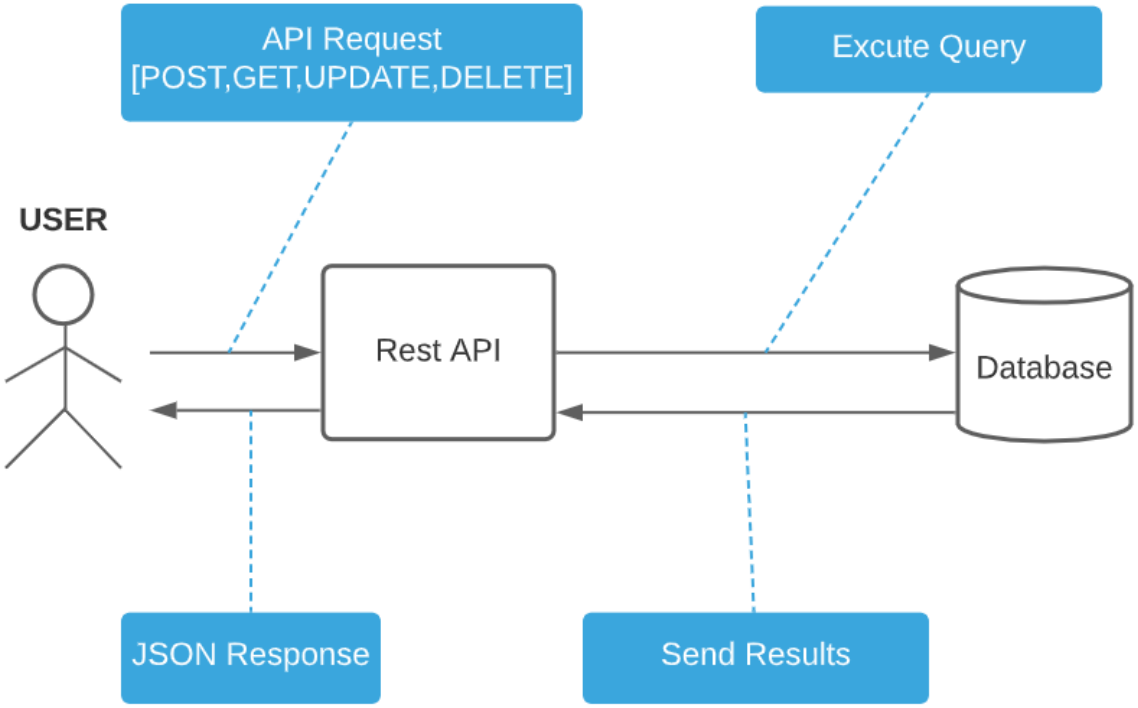

Figure 2.3: Overview of the REST API Architecture

Figure 2.3 represents a general architecture of our REST API. User can make an HTTP request (GET, POST, PUT or DELETE) through a mobile app, desktop app or web app, ext then the REST API interrogate the data base then the REST API sends HTTP response to the user in Json format.

- **GET** used to request for information such as profile information.
- **POST** used to create new information or send confidential information such as passwords.
- **PUT** used to update existing data in the database.
- **DELETE** used for deleting contents from database such as posts.

5.1 Recommendation Task Design

In order to get the meal slate, the user send GET request to the REST API, next the recommender agent will interrogate the data base to get the user current state, then the recommender agent will generate the meal slate for the user and send it to the user through an HTTP response in json format. Figure 2.4 shows the recommendation task design in the REST API.

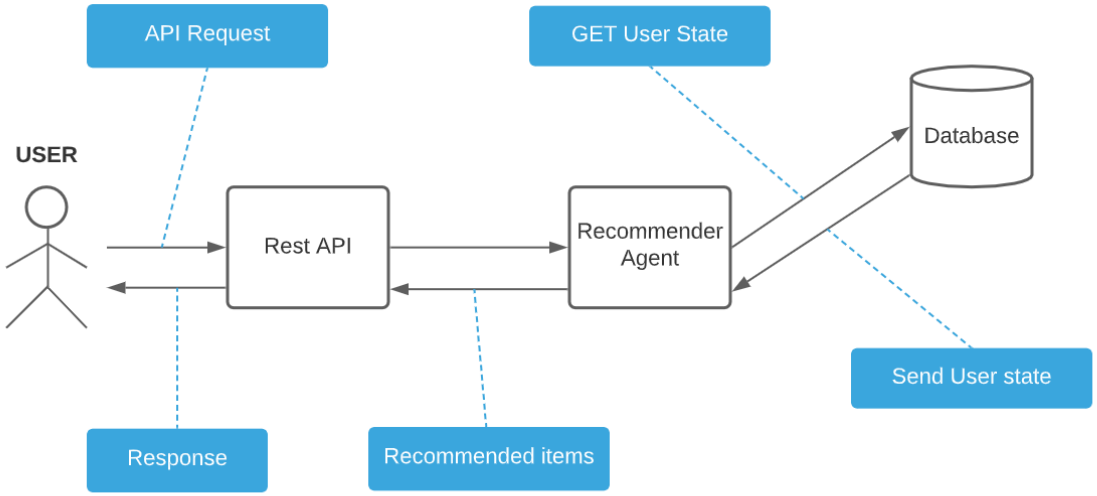

Figure 2.4 : the recommendation task design in the REST API.

5.2 Use Case Diagram

The REST API that we designed contains a lot of interesting functionalities in healthcare sector such as booking appointments, search for doctors and food recommendation. We also provide functionalities for doctors by enabling them to create profile and manage patients appointments online. The use case diagram in the figure below describes all the available functionalities in our REST API.

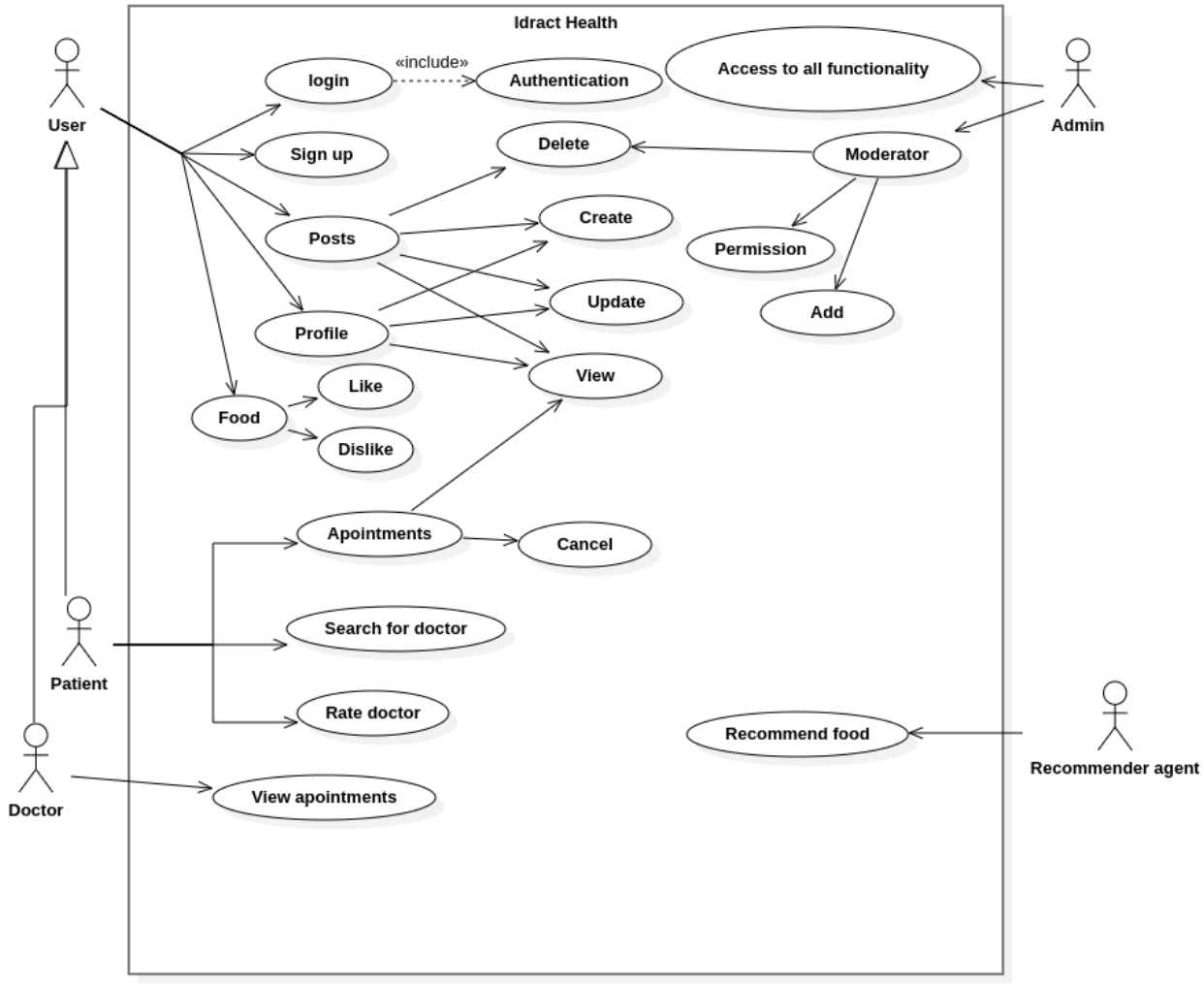

Figure 2.5 : the use case diagram

5.3 Class Diagram

Figure 2.6 represent a class diagram to show how the data are stored in the database and also the links between Entities in the database.

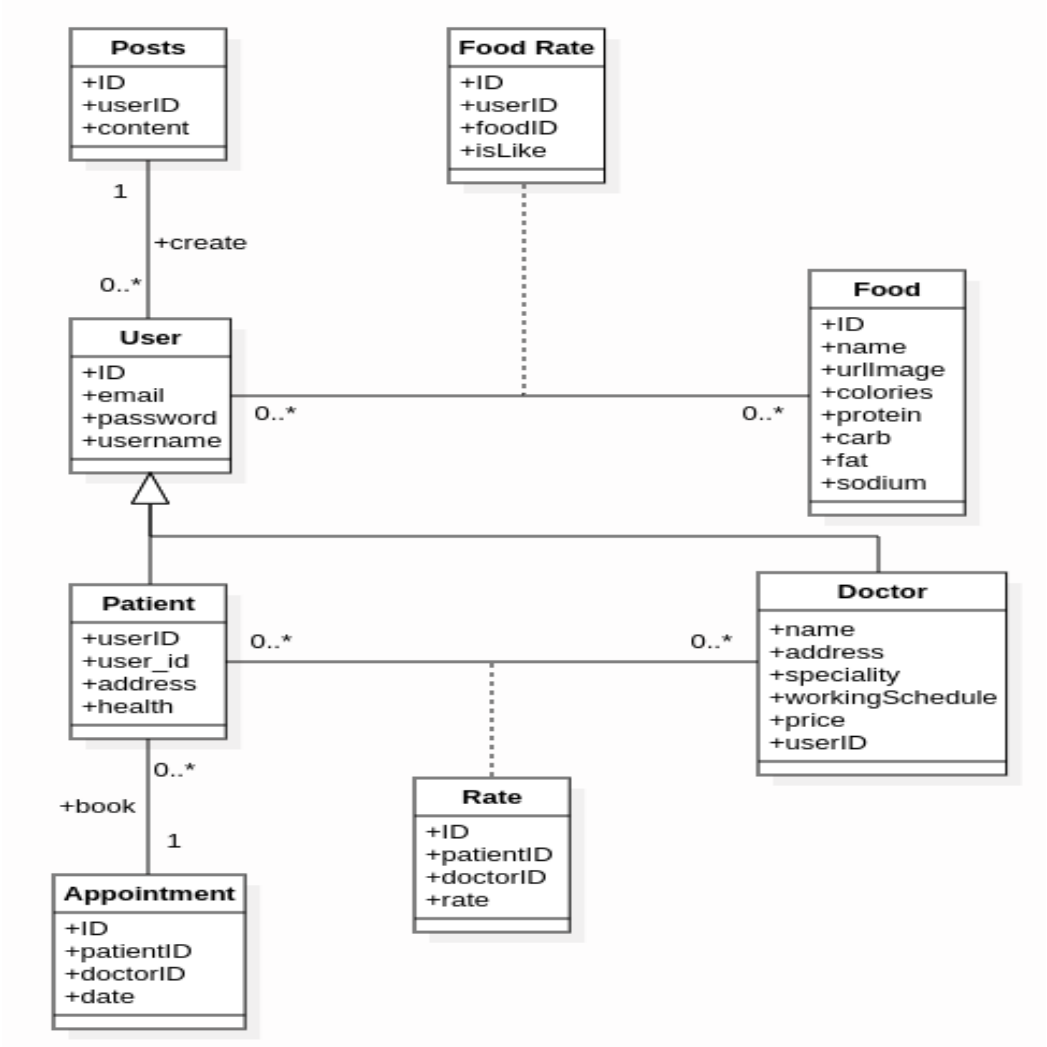

Figure 2.6 class diagram

5.4 Activity Diagrams

We are going to use UML activity diagram to describe the dynamic aspects of our system and the flow between activities. We have an activity diagram for the admin activities and another one for the users activities.

Admin activity diagram

Figure 2.7 represents the flow of the admin activities such as how to delete users or grant permissions for accounts.

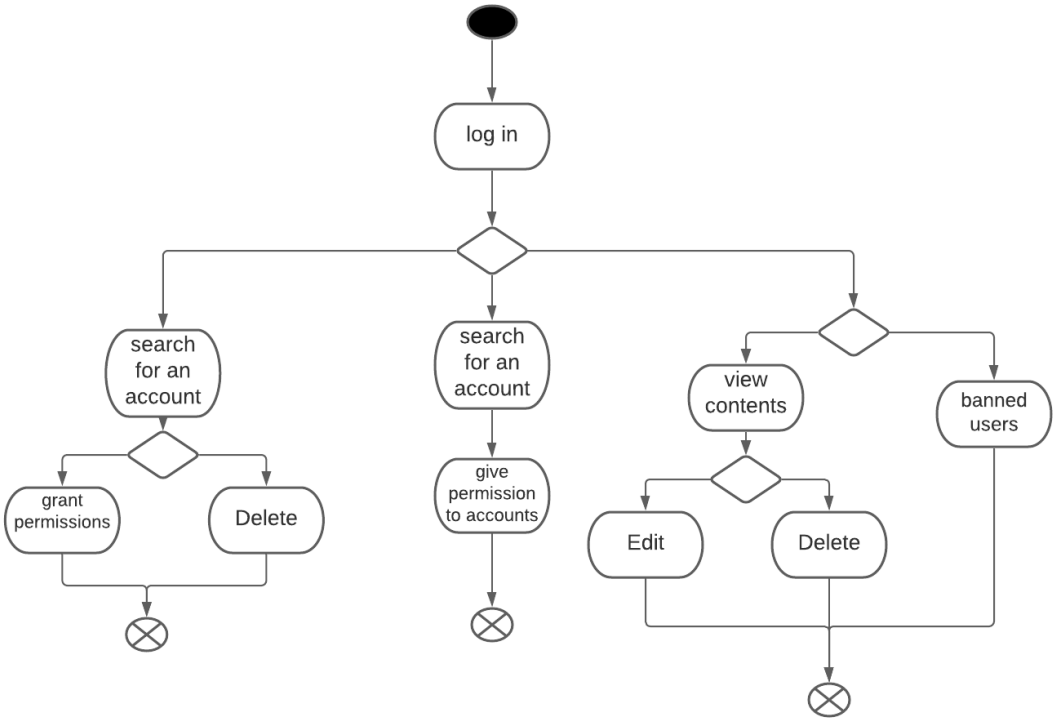

Figure 2.7 : Admin activity diagram

Activity Diagram for Users

Figure 2.8 represent the flow of activities that can be done by users such as sign up, log in and create appointments.

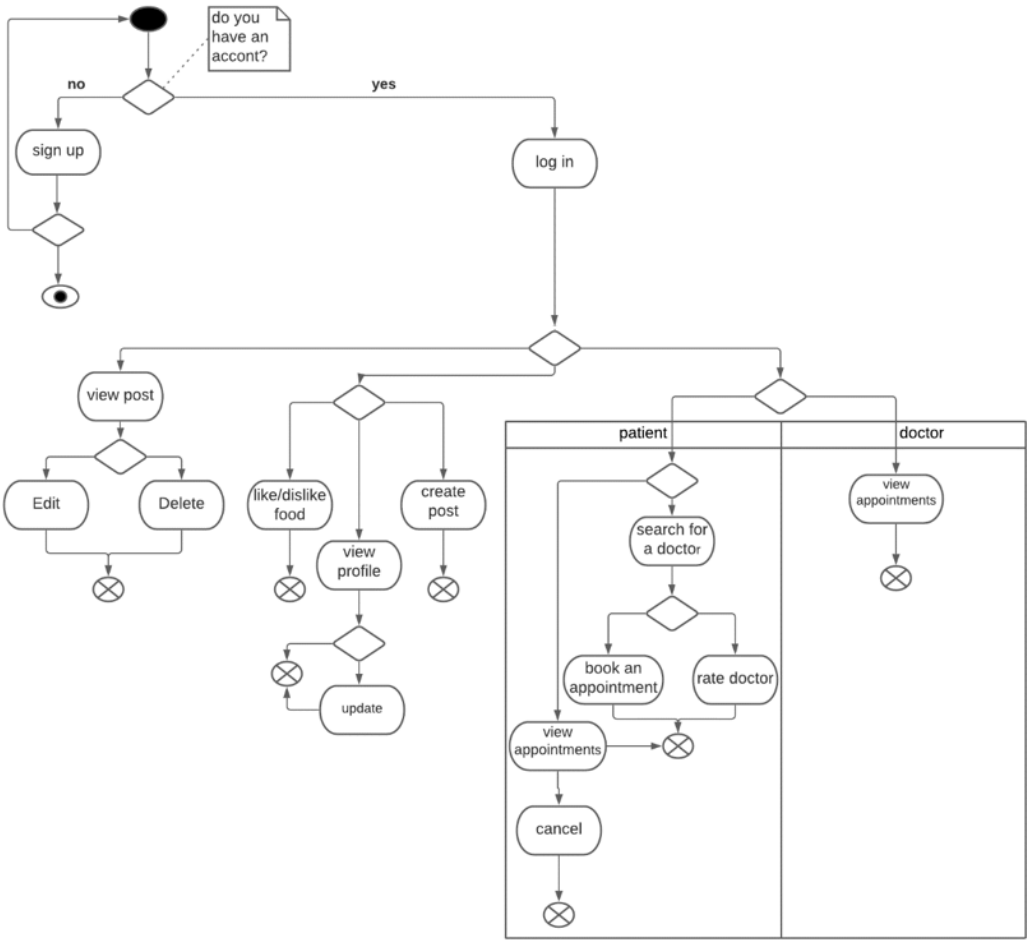

Figure 2.8 : Activity Diagram for Users

5.5 Sequence Diagrams

In this section we are going to describe the sequence of the interactions that happens between user and the system of few functionalities such as the recommendation task, sign up functionality and so on using the sequence diagram.

Sequence Diagram for the Recommendation

Figure 2.9 bellows shows the interaction between the recommender agent and the user and also describes how the recommender agent online training, which starts by the agent takes an action (recommendation slate), then the application shows the recommended items (food) to the user then the user sends his feedback, next the user feedback used to calculate the error of the recommendation and then the error will be used to update the DQN network weights(section **4.2.2**) in order to increase the quality of the recommendation.

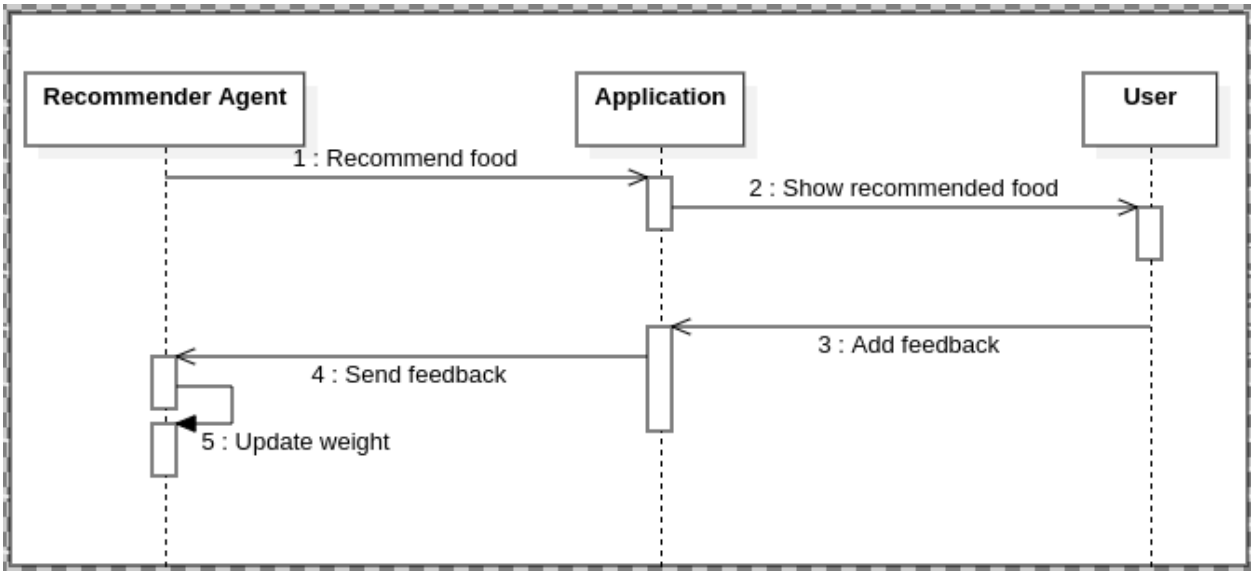

Figure 2.9: Sequence Diagram for the Agent training

Sequence Diagram for Sign Up

Figure 2.10 describes the sequence of interactions user—system in the sign up functionality, which starts by the user entering his information then it stored in the database then the system sends a verification email to the user to confirm his identity.

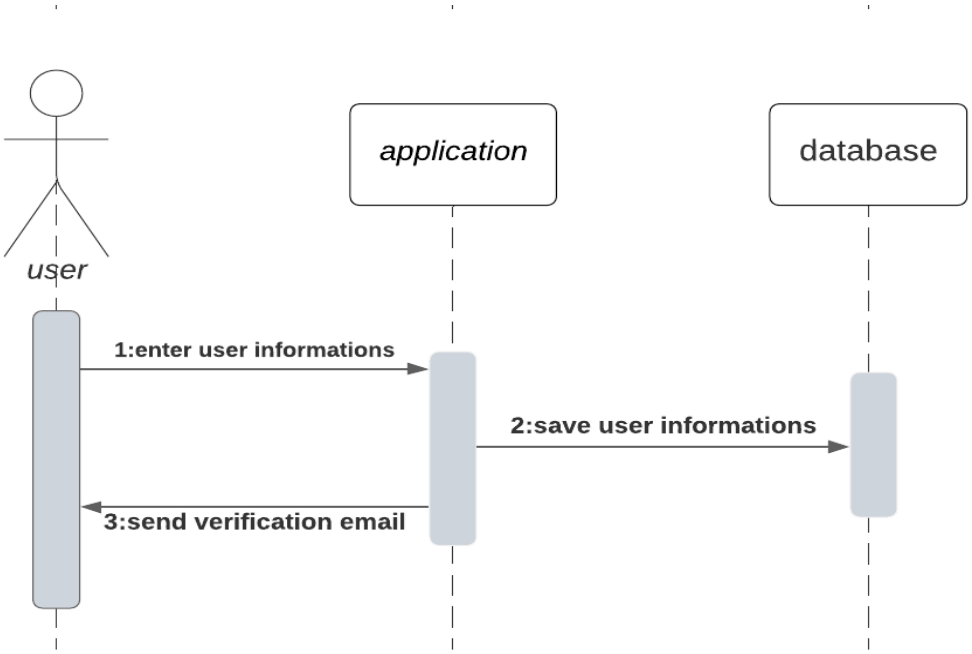

Figure 2.10 : Sequence Diagram for SignUp

Sequence Diagram for viewing appointments

Figure 2.11 describes the sequence of interactions between doctor—system of an interesting functionality for the

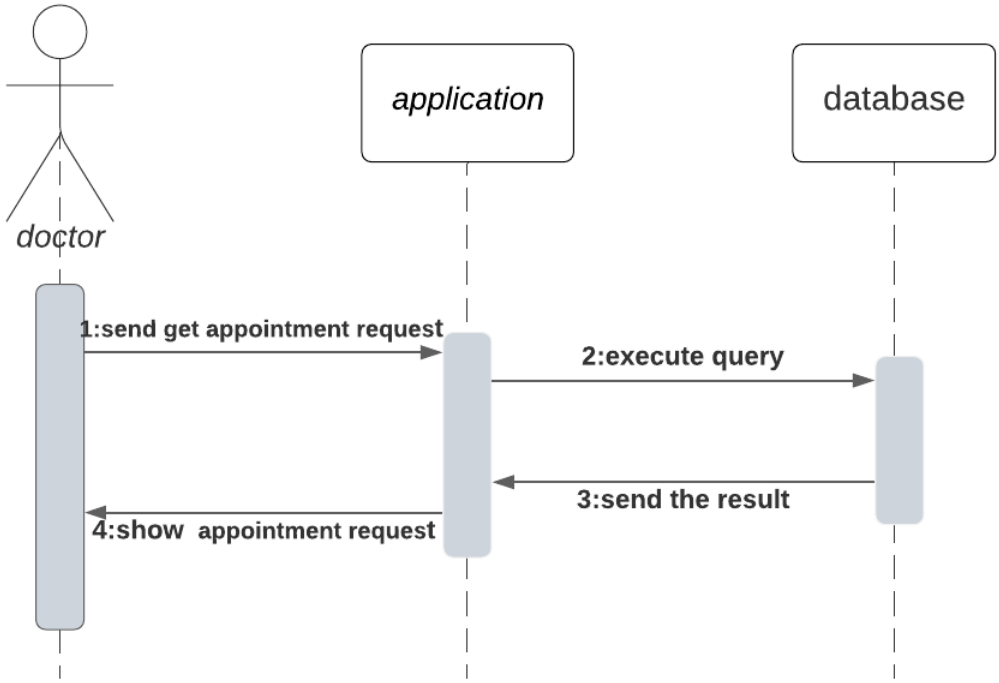

Figure 2.11 : Sequence Diagram for viewing appointments

doctors which enable them to view the list of patients that books an appointment with them.

Sequence Diagram for booking an appointment

Figure 2.12 describes the sequence of interactions between patient—system of an interesting functionality for the patients which enable them to look for doctors and book an appointments with them.

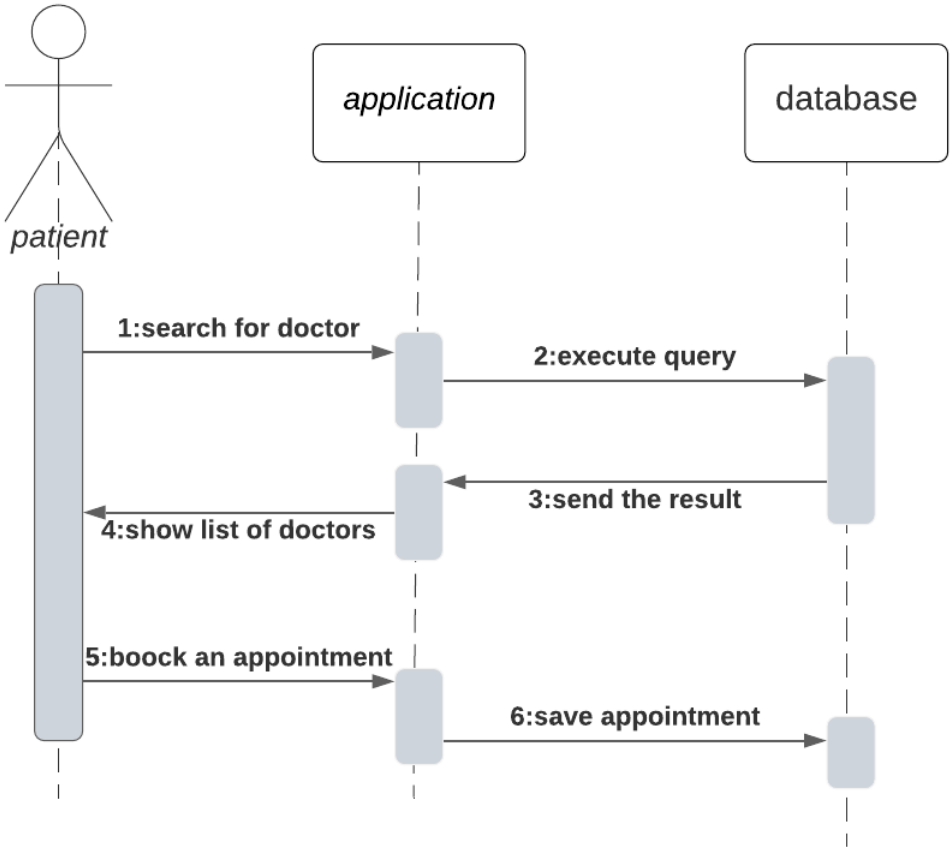

Figure 2.12 : Sequence Diagram for booking an appointment

Sequence Diagrams for Posting system

Figures 2.13 and 2.14 represent sequence diagram for creating a post and viewing posts. The posting system is a very interesting functionality which helps users to share their experiences with others and also learn from the experiences of others for example a user share his experience of dealing with chronic disease such as diabetes and also doctors can post healthcare advises such as dealing with hypertension.

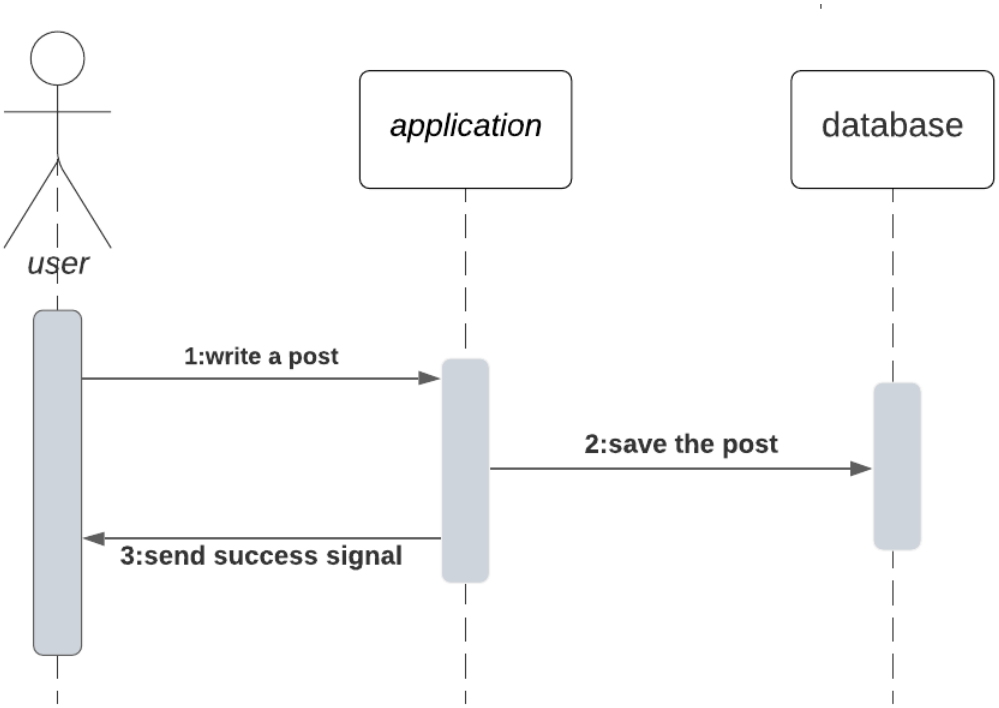

Figure 2.13 : Sequence Diagrams for Posting system

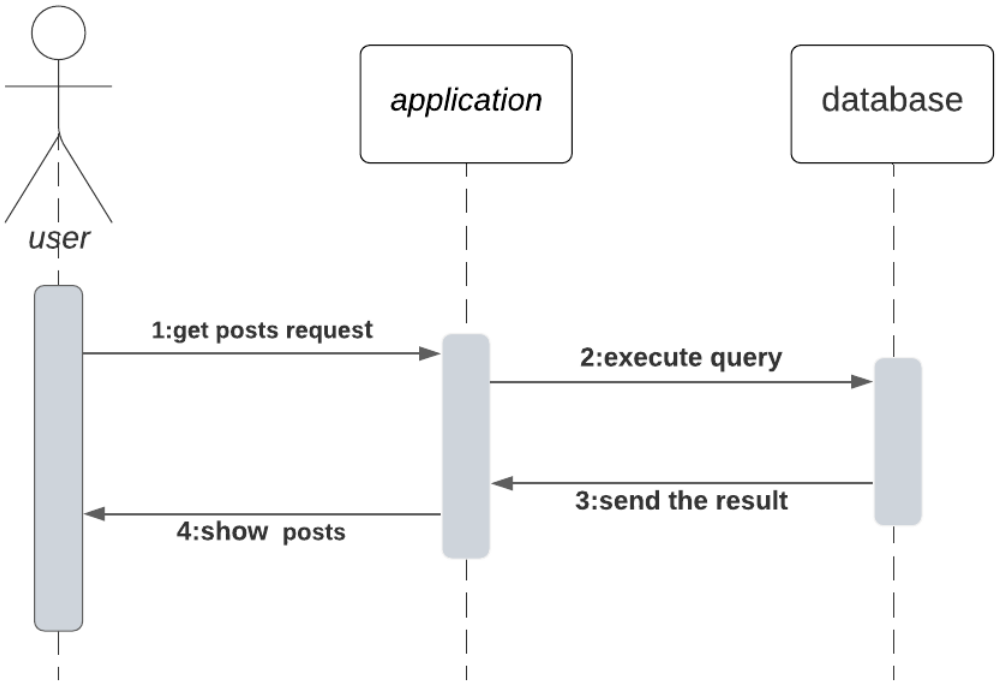

Figure 2.14 : Sequence Diagrams for Posting system

3. Conclusion

In this chapter we addressed the importance of the recommendation systems as well as the importance of designing robust recommender systems which should take care of users interests in the long run, we also addressed the importance of defining a good state representation, which is critical for decision making task. System design is important to accelerate the development cycle, make the system easy to debug and facilitate adding extension to the system.

Chapter 3 : Realisation

1. Introduction

In this Chapter, we first describe the hardware and software tools used in developing the proposed system, then we explain the implementation that we have conducted to verify the effectiveness of our approach

2. Development Environment and Tools

2.1 Hardware Platform:

The implementation of the application is carried out on a laptop having the following characteristics:

- Machine: hp
- Processor: Intel (R) Core (TM) i3-6006U
- Frequency: 2.00GHz
- RAM: 4.00 GB
- Operating system: Linux

2.2 Software Platform:

This section presents the tools used for the development of the proposed system.

Flutter: is a free and open-source mobile UI framework created by Google and released in May 2017. In a few words, it allows you to create a native mobile application with only one codebase. This means that you can use one programming language and one code base to create two different apps (for iOS and Android).

Flutter consists of two important parts:

- An SDK (Software Development Kit): A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).

- A Framework (UI Library based on widgets): A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.[45]

3. Technology that used

Our application is developed by rest frameworks and flutter

-REST frameworks is acronym for representational State Transfer. It is architectural style for distributed hypermedia systems and was first presented by Roy Fielding in 2000 in his famous dissertation. REST framework is a powerful and flexible toolkit for building Web APIs. Its main benefit is that it makes serialization much easier.

-Flutter is a cross-platform UI toolkit that is designed to allow code reuse across operating systems such as iOS and Android, while also allowing applications to interface directly with underlying platform services ,The goal is to enable developers to deliver high-performance apps that feel natural on different platforms, embracing differences where they exist while sharing as much code as possible.

Some reasons to use REST framework:

- Its Web-browsable API is a huge usability win for your developers.
- Authentication policies include packages for OAuth1 and OAuth2.
- Serialization supports both ORM and non-ORM data sources.
- It’s customizable all the way down. Just use regular function-based views if you don’t need the more powerful features.
- It has extensive documentation and great community support.
- It’s used and trusted by internationally recognized companies including Mozilla, Red Hat, Heroku, and Eventbrite.[46]

4. Application handling:

our system provides several services to administrator and users, and for fulfilled this task we need:

• First, the admin side
• Then, the user side

Admin graphic user interface

Once you've logged in to your user account, you'll see the administration user interface (adminui). admin can proforme allot of actions like:

- Create and configure users accounts and profiles, including creating and assigning permission and roles to users.
- Configure and schedule publishing sessions.
- Create and manage workflow processes.

The figure below resum admin action:

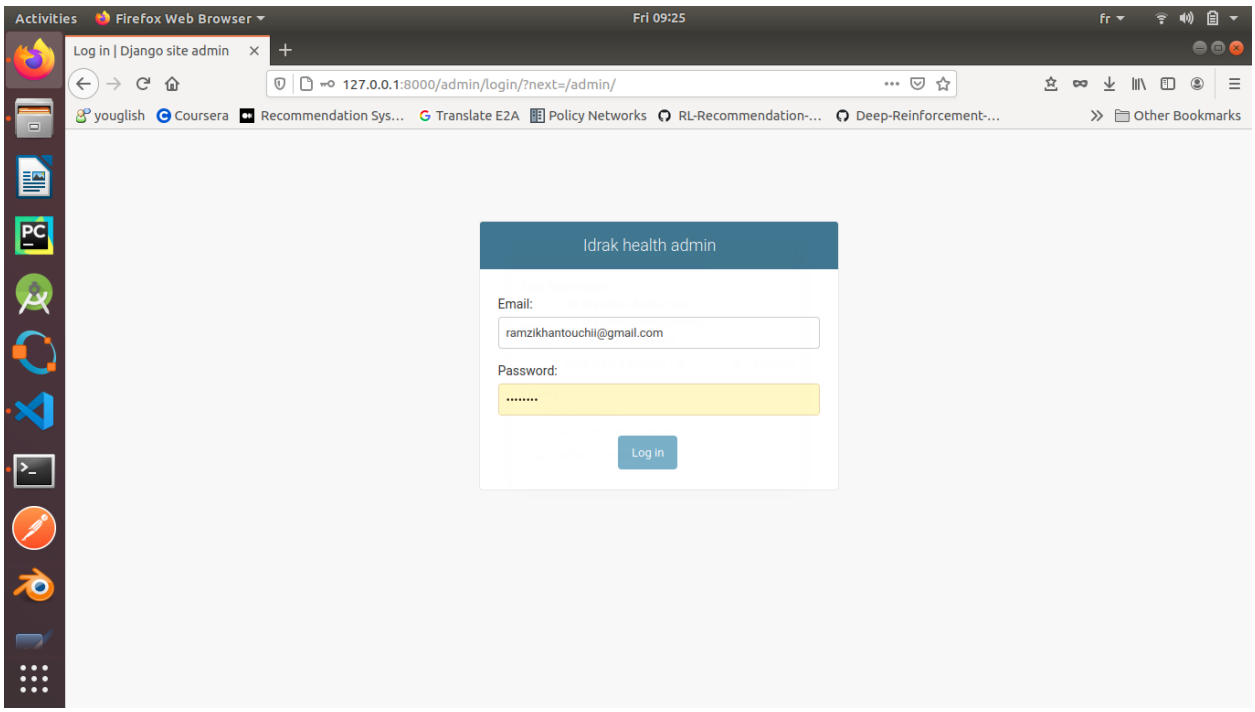

FIGURE 3. 21 ADMIN LOG IN

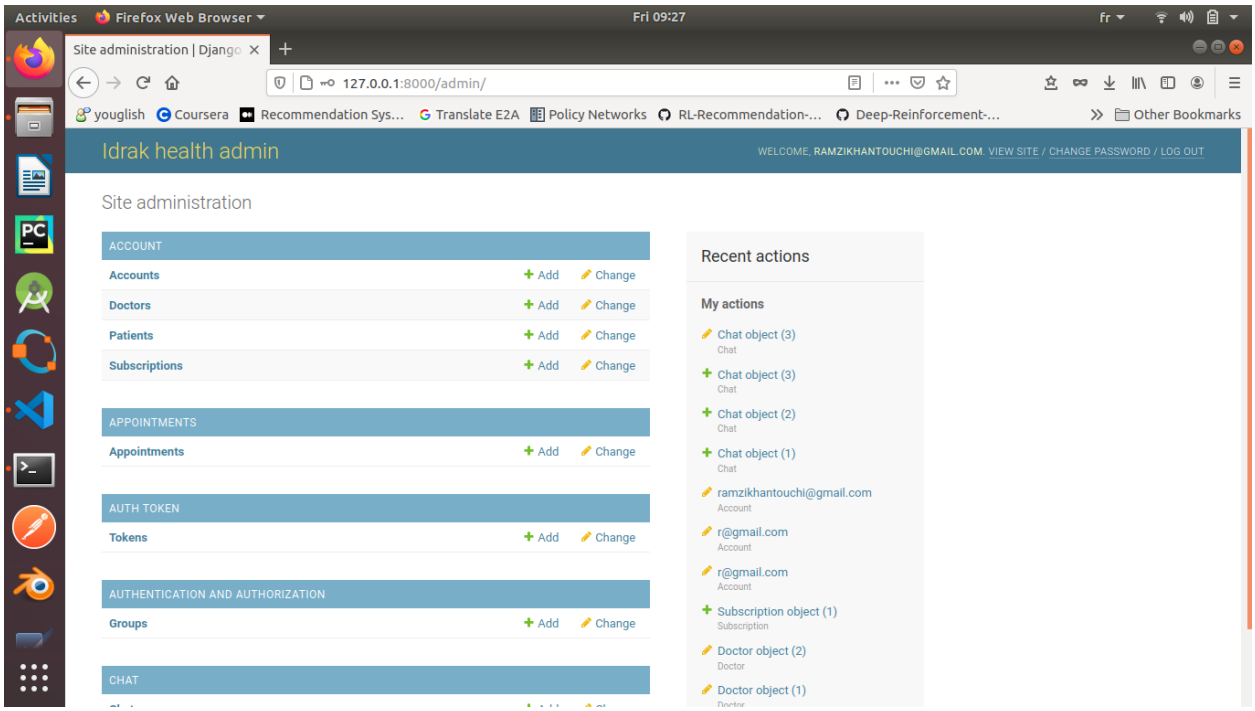

Figure 3. 22 dashbord of admin account

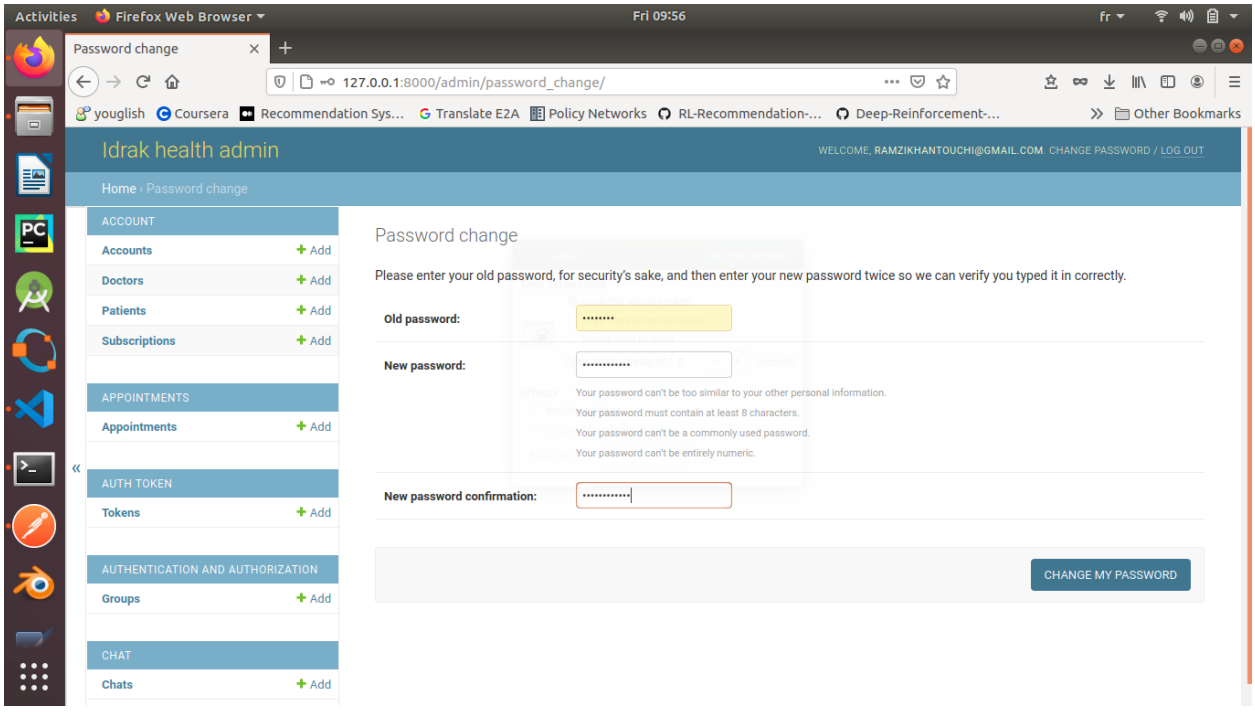

Figure 3. 23 change password

Admin manage account

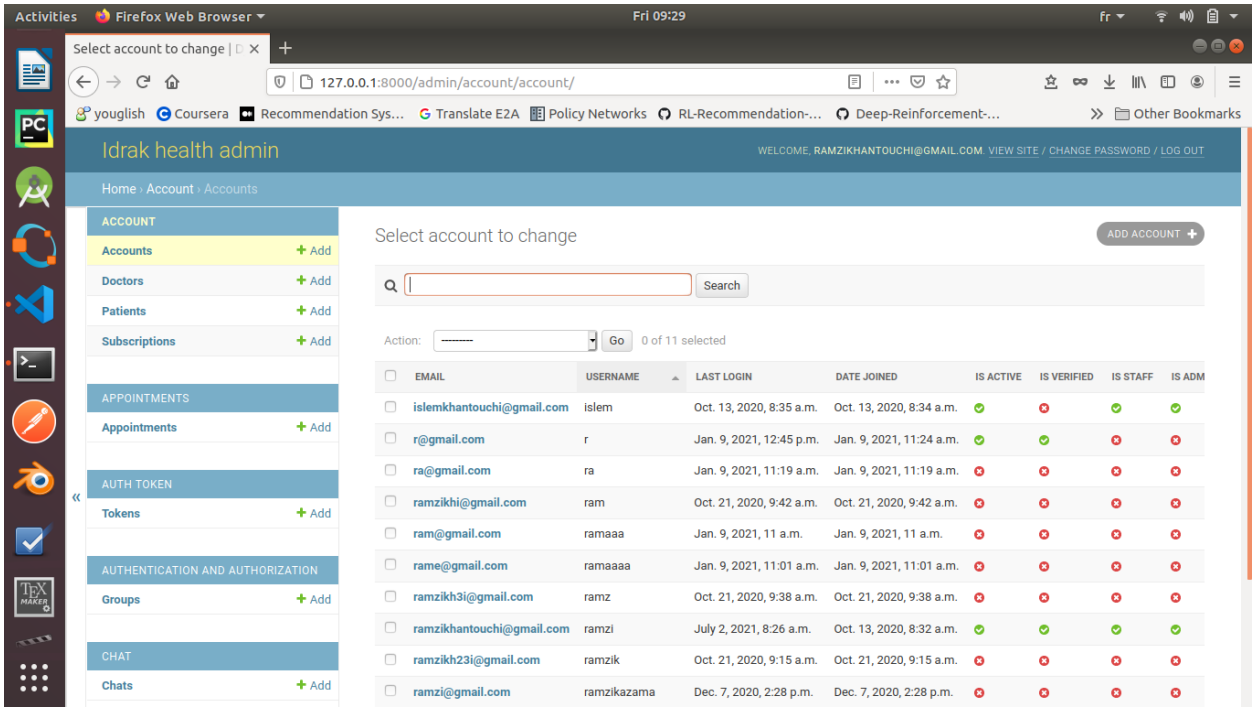

Figure 3. 24 acounts managment

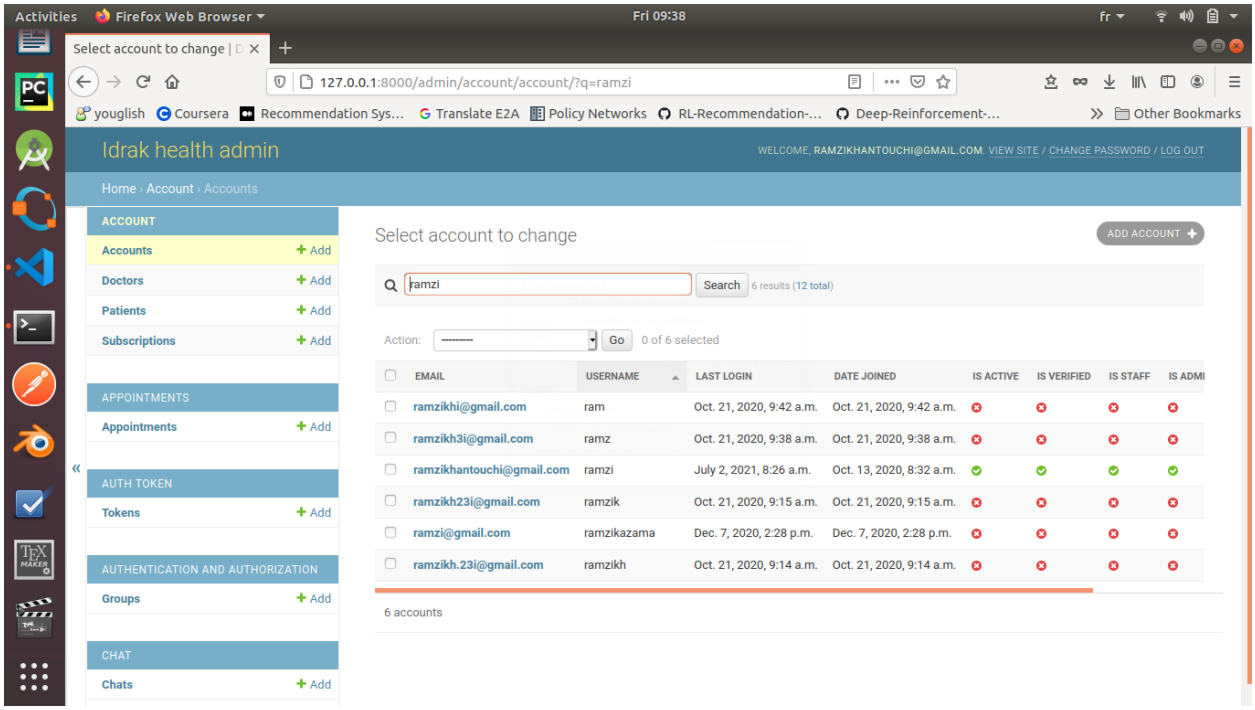

Figure 3. 25 search for accounts

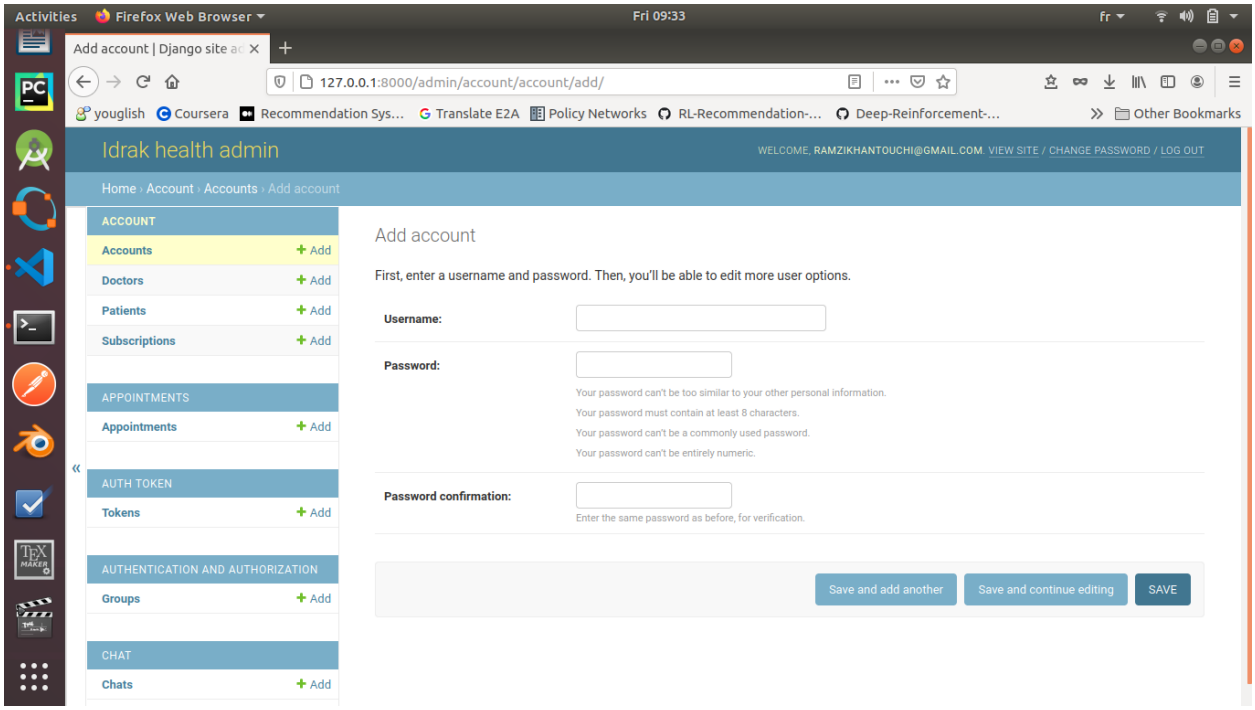

Figure 3. 26 create account

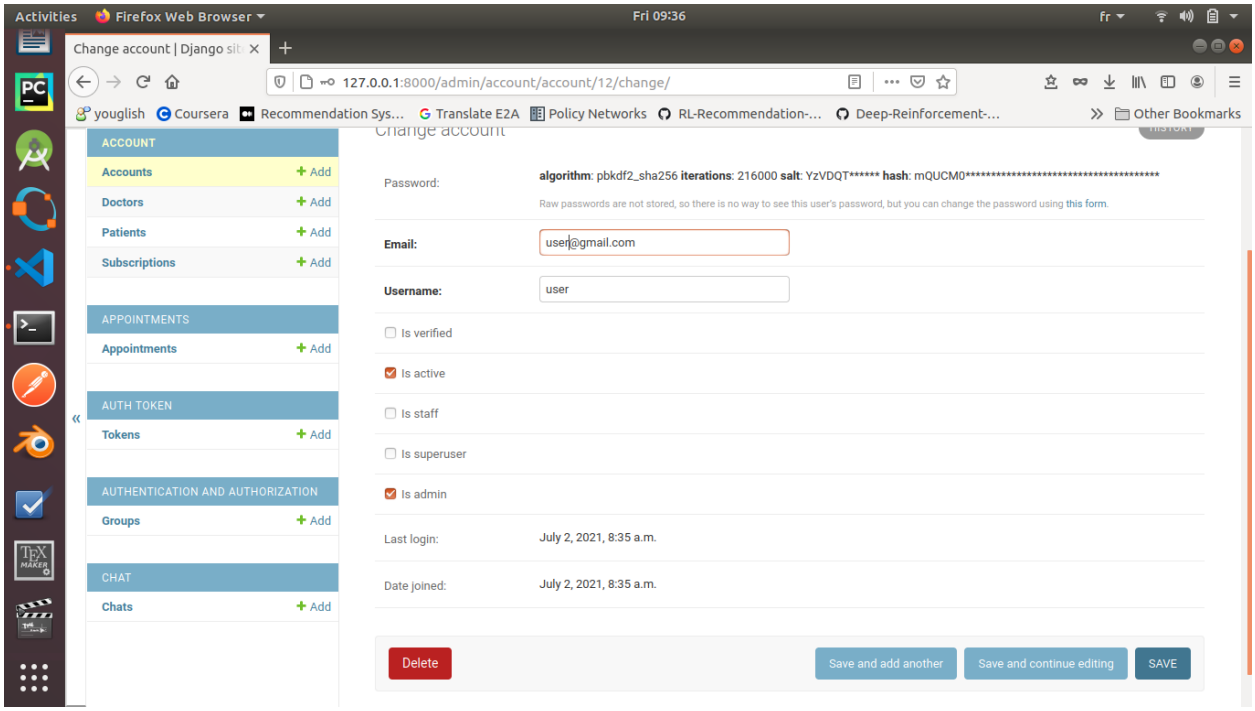

Figure 3. 27 create account with permission graint

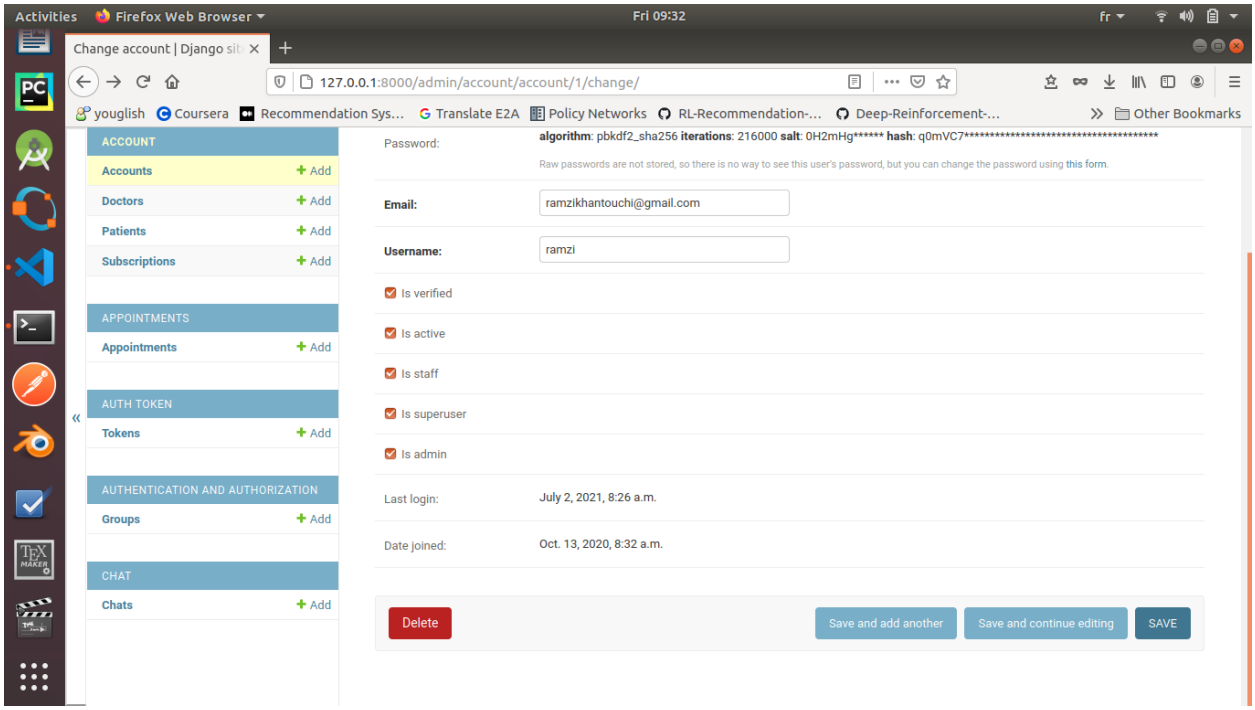

Figure 3. 28 save / edit/ delete option

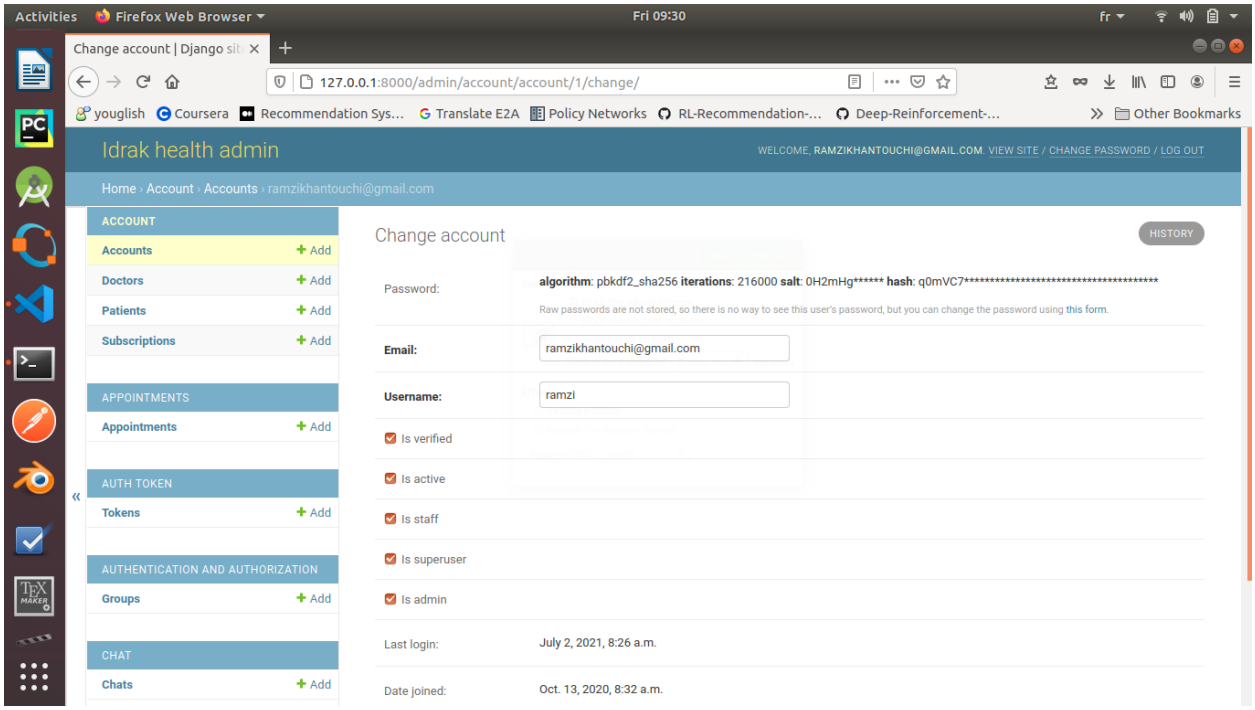

Figure 3. 29 account details

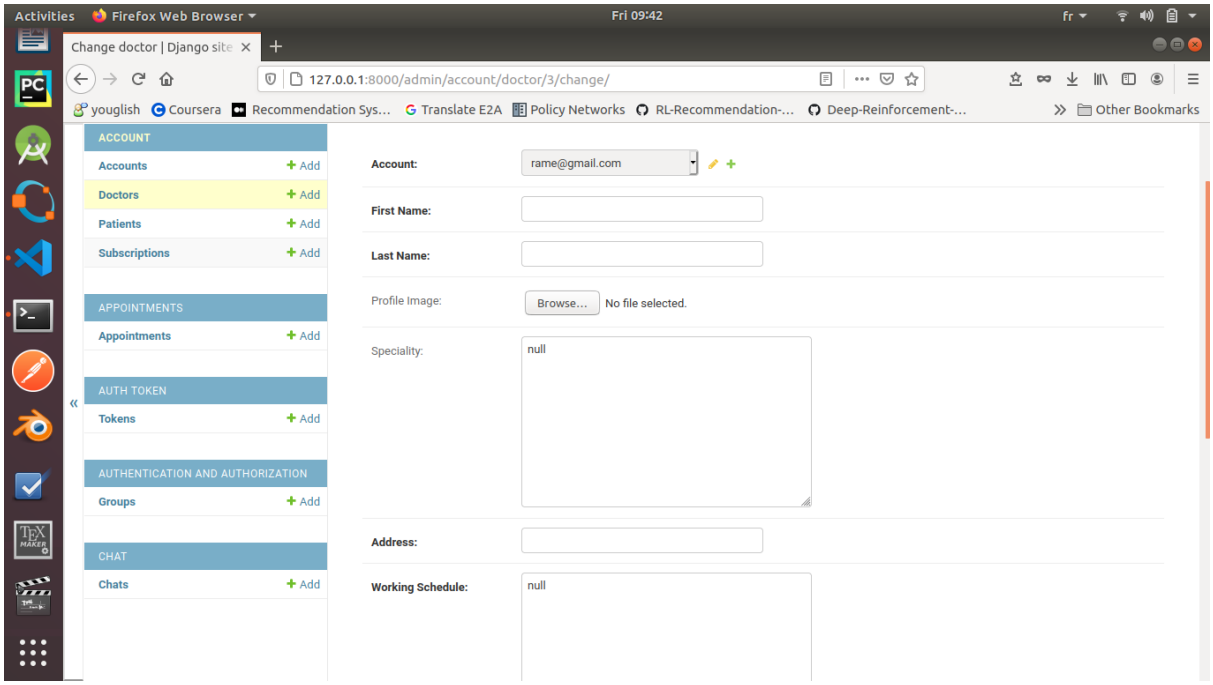

Figure 3. 30 doctor account information_1

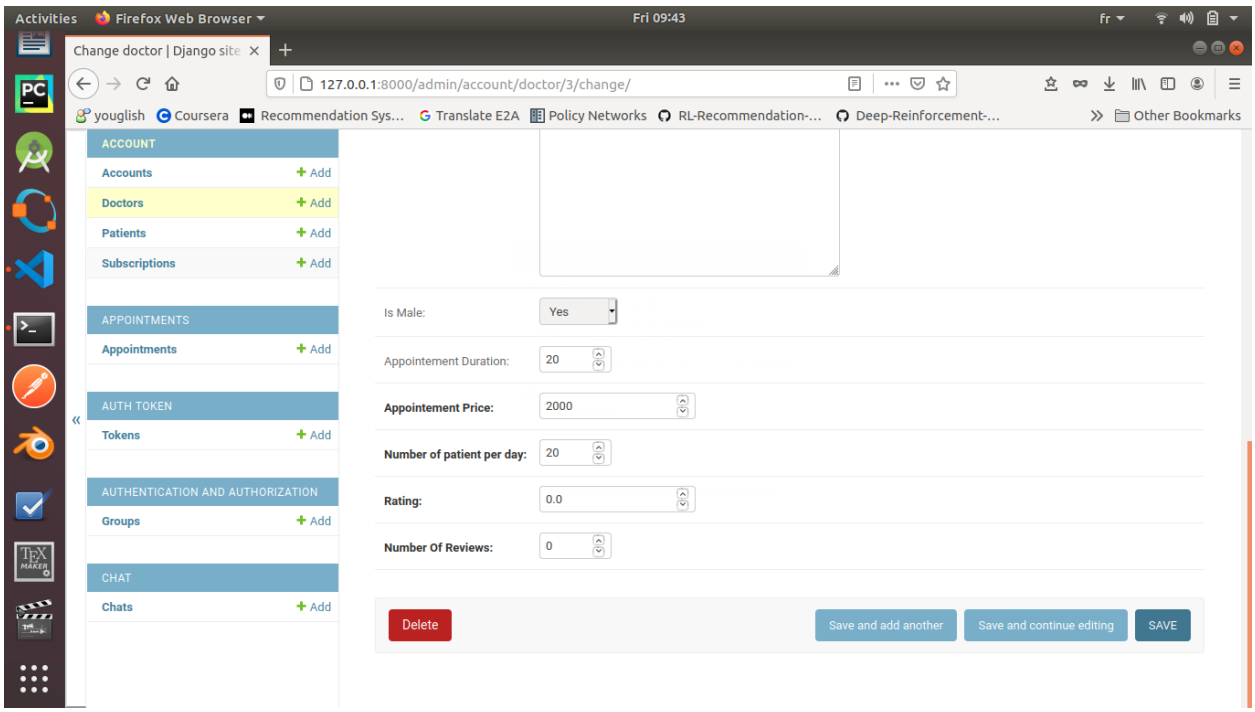

Figure 3. 31 doctor account information_2

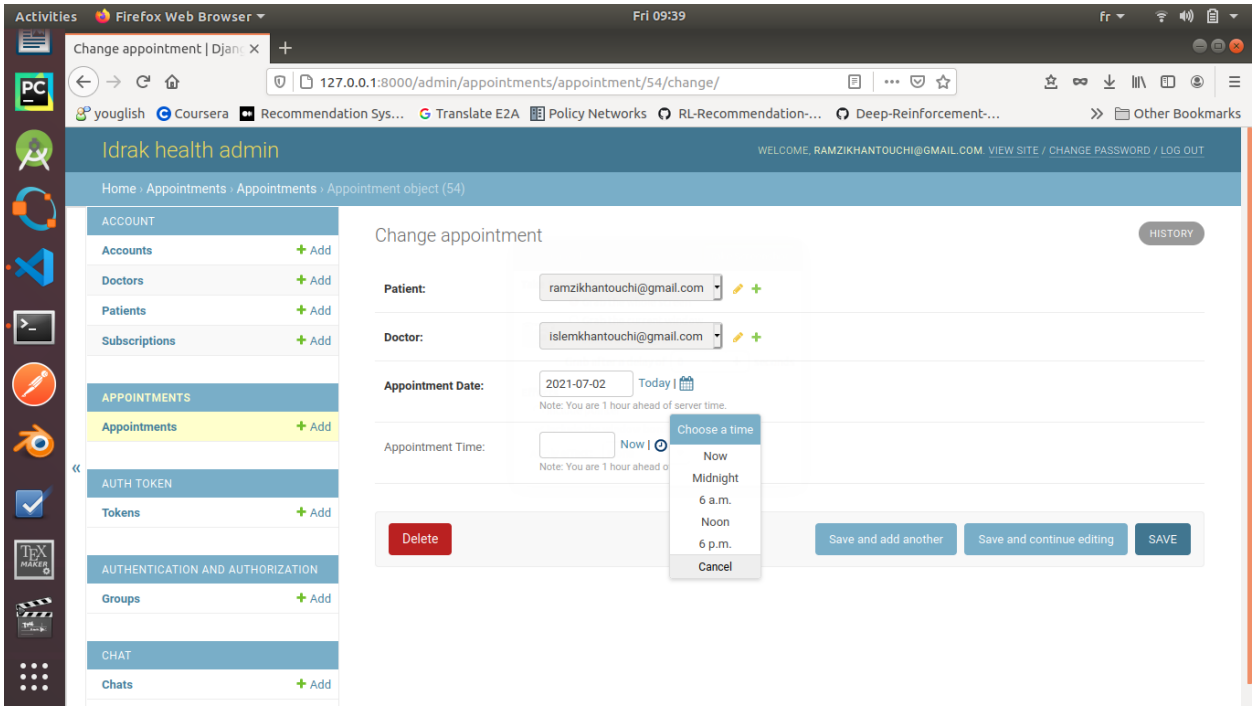

Figure 3. 32 patient appointment

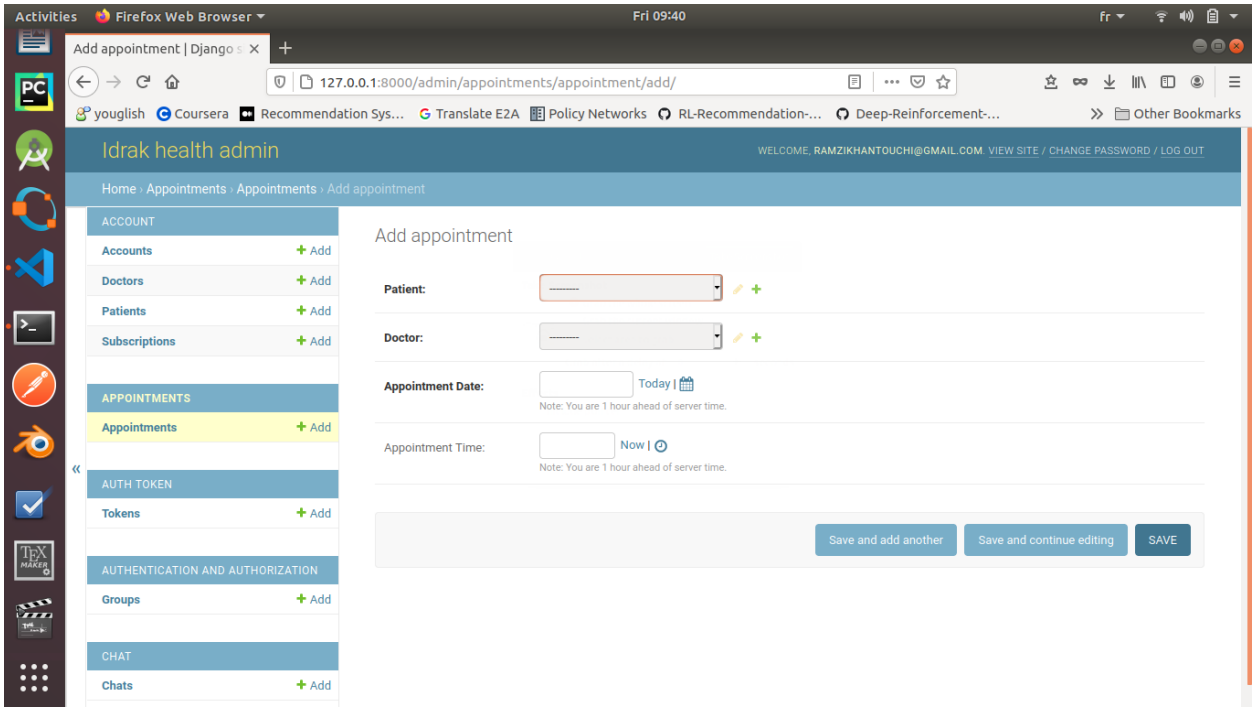

Figure 3. 33 add appointment

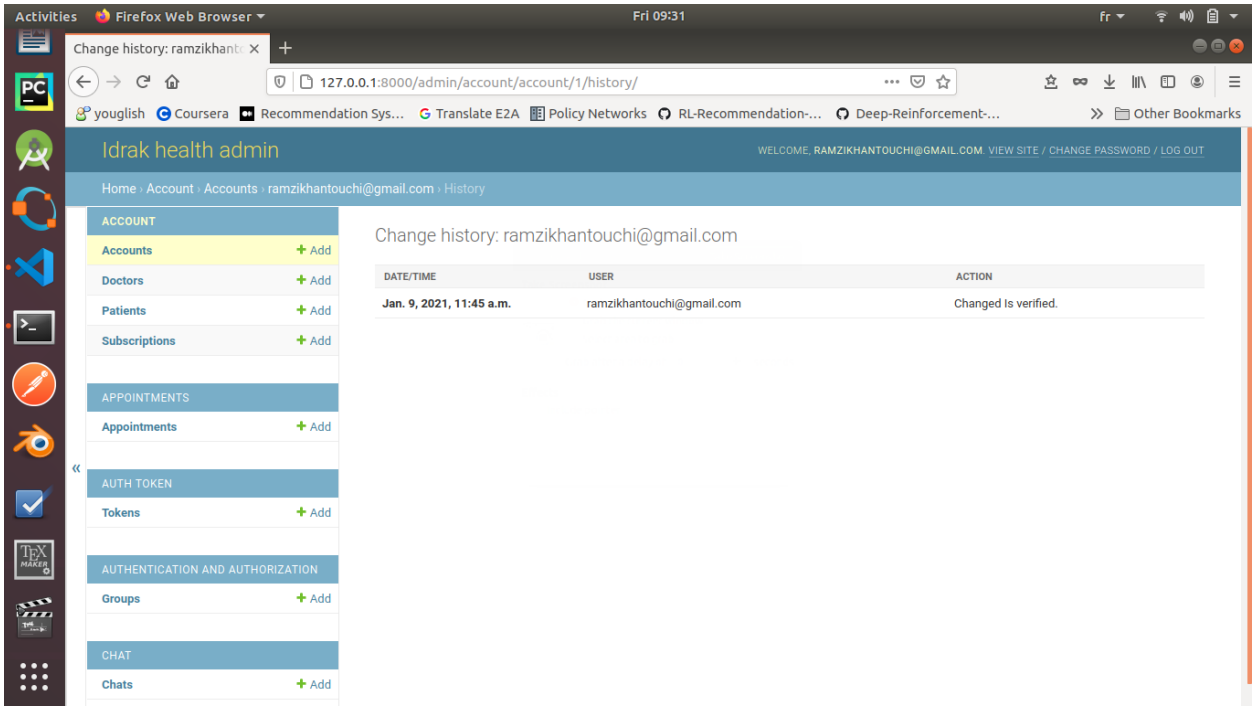

Figure 3. 34 user history

Client user interface

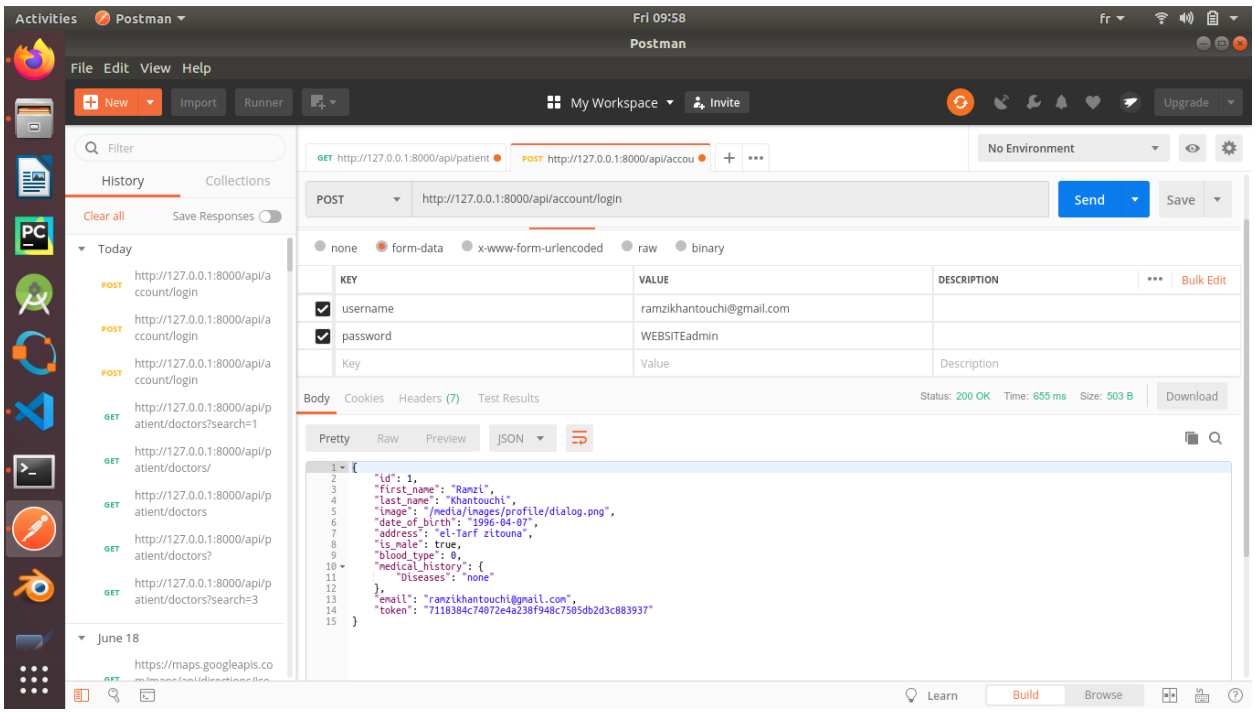

Figure 3. 35 log in request

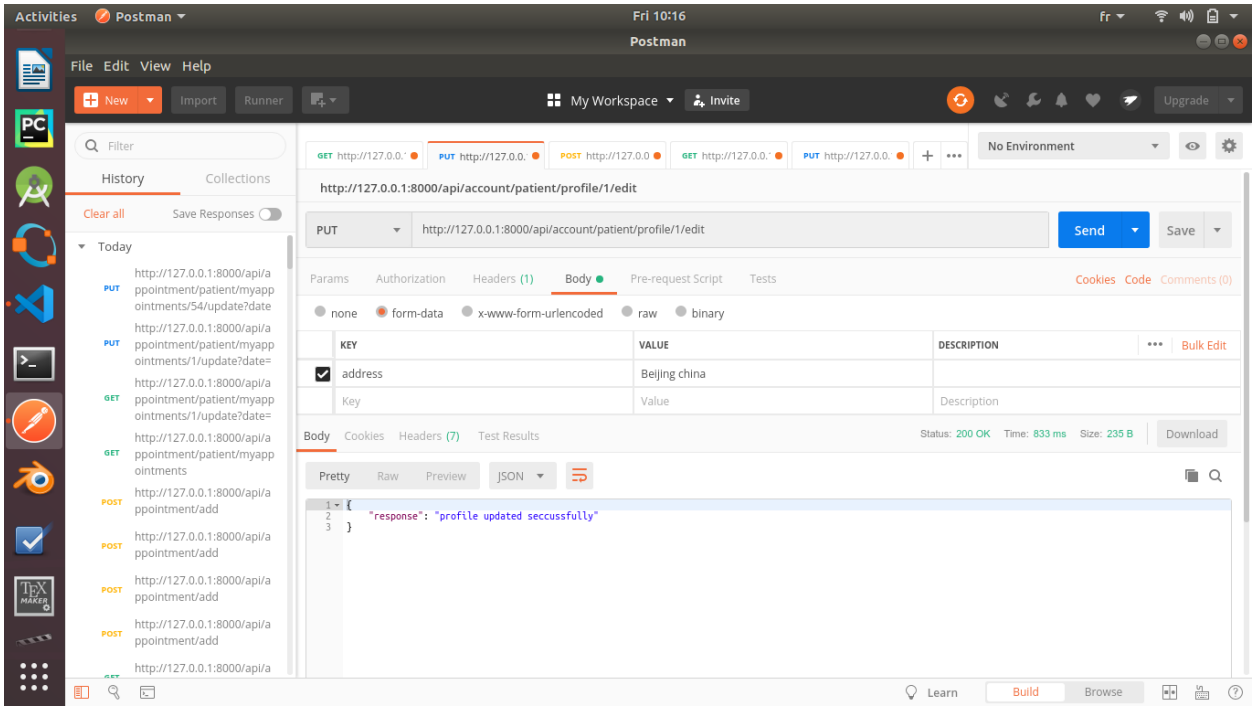

Figure 3. 36 profiel edit

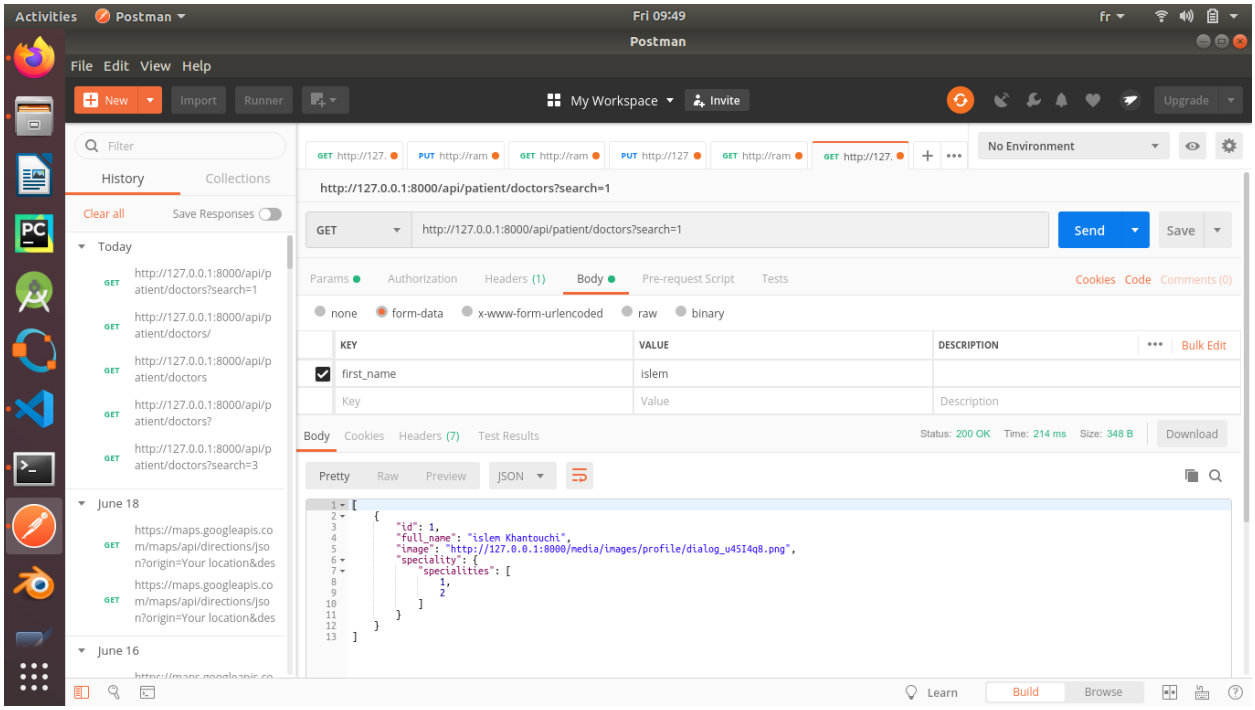

Figure 3. 37 search for a doctor by name

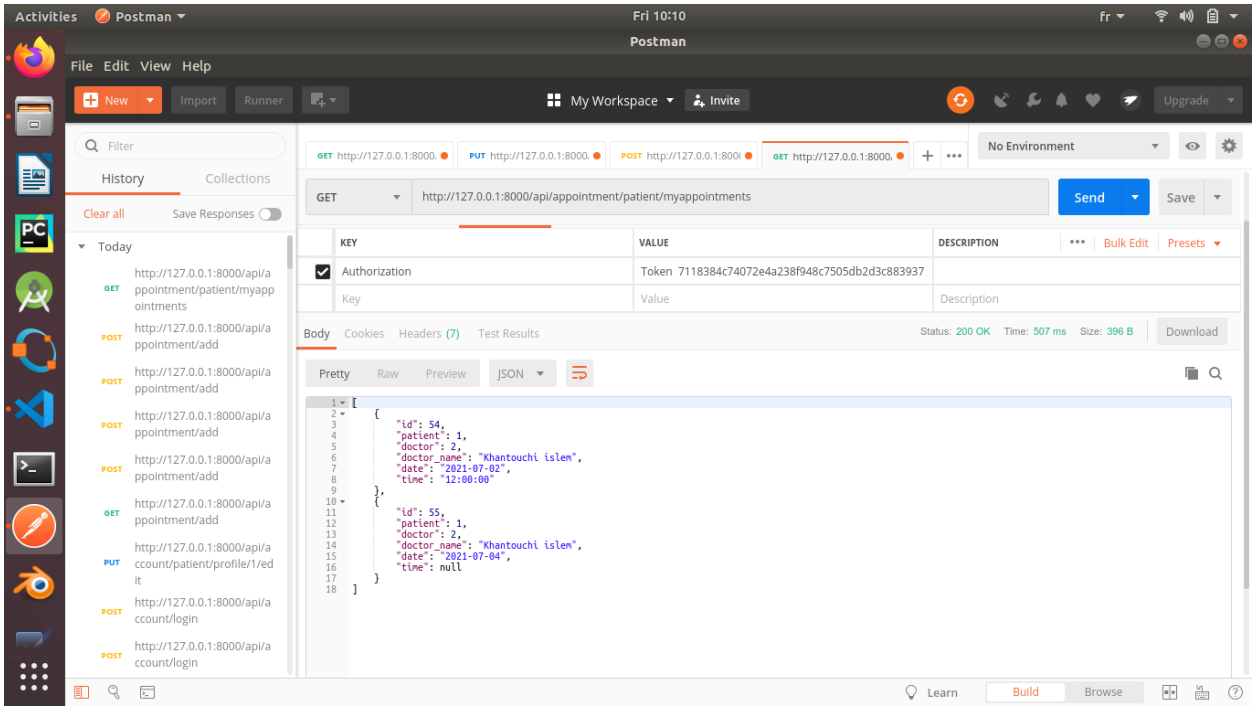

Figure 3. 38 get the appointments list

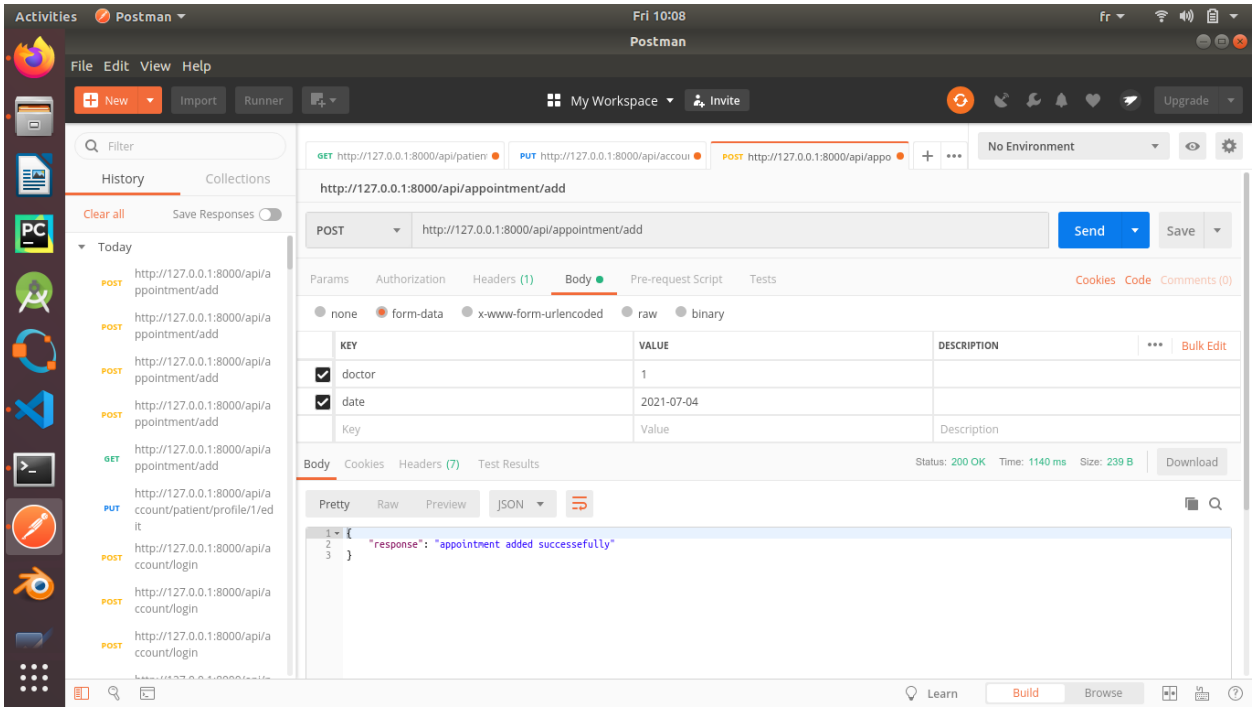

Figure 3. 39 create an appointment

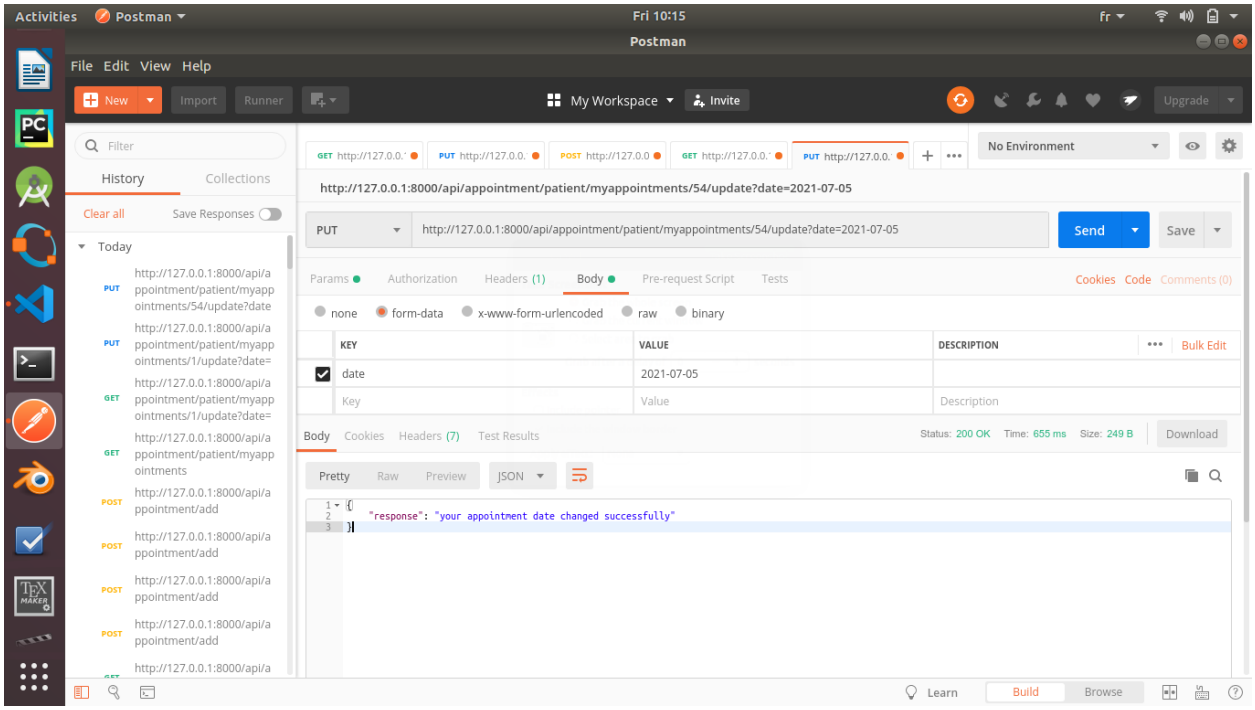

Figure 3. 40 change an appointment date

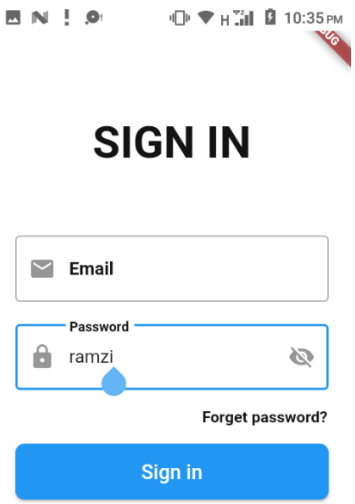

Figure 3. 41 sign in page

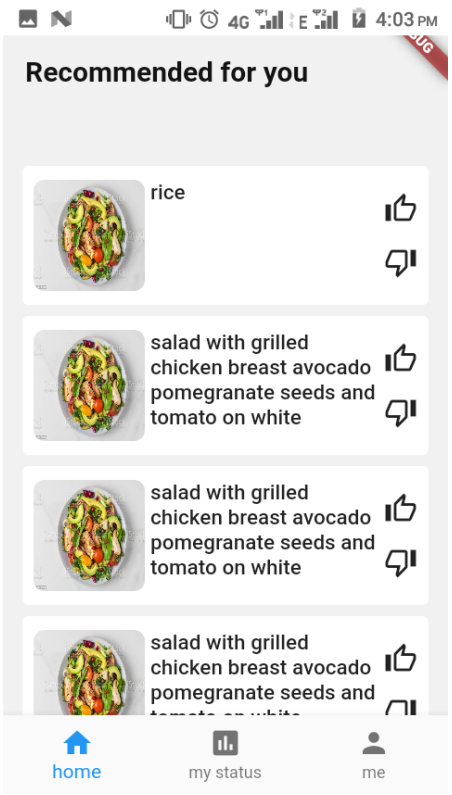

Figure 3. 42 home page

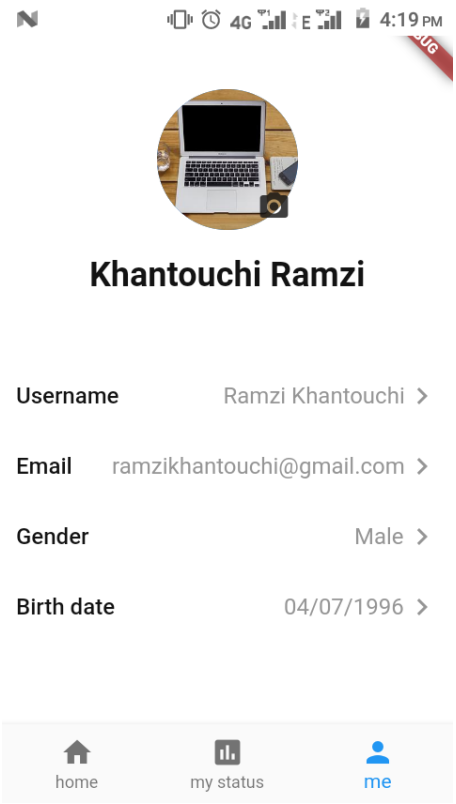

Figure 3. 43 profiel page

5. Conclusion

In this chapter, we have seen the criteria on which we chose the programming language, the general interface and the various functionalities of the system, as well as the obtained results.

Conclusion and Perspectives

This thesis proposes an online model-free reinforcement learning framework for food recommendation.

The aim of the proposed method is to enhance the quality of recommendation over time. This framework introduces three main components; the first one uses a deep learning technique called Doc2vec to encode user state. Then, a value based reinforcement leaning algorithm called Deep Q Learning is applied. It takes the user encoded state as an input, and it provides the best candidate item to be recommended. Finally, the third component generates a Top-N recommended items using KNearest Neighbors (knn) algorithm by taking the best candidate and return k neighbors of that candidate. The experiment results indicates that the quality of the recommendation given by the proposed framework is improving over time.

In a future work, we plan to enhance the quality of the user state representation by incorporating other useful information.

References

[1] Moradi P, Ahmadian S. A reliability-based recommendation method to improve trust-aware recommender systems. Expert Systems with Applications. 2015;42: 7386–7398. doi:10.1016/j.eswa.2015.05.027

[2] Shao B, Li X, Bian G. A survey of research hotspots and frontier trends of recommendation systems from the perspective of knowledge graph. Expert Systems with Applications. 2021;165: 113764. doi:10.1016/j.eswa.2020.113764

[3] Patro SGK, Mishra BK, Panda SK, Kumar R, Long HV, Tuan TM. Knowledge-based preference learning model for recommender system using adaptive neuro-fuzzy inference system. Journal of Intelligent & Fuzzy Systems. 2020;39: 4651–4665. doi:10.3233/JIFS-200595

[4] Herlocker J, Konstan JA, Riedl J. An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. Information Retrieval. 2002;5: 287–310. doi:10.1023/A:1020443909834

[5] Reinforcement Learning: An Introduction Richard Sutton and Andrew G. Barto 2018

[6] David Silver (phd thesis) Reinforcement Learning and Simulation -Based Search in Computer GO

[7] Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. Nature **529,** 484–489 (2016). https://doi.org/10.1038/nature16961

[8] Silver, D., Schrittwieser, J., Simonyan, K. et al. Mastering the game of Go without human knowledge. Nature **550,** 354–359 (2017). https://doi.org/10.1038/nature24270

[9] Paul Resnick and Hal R Varian. Recommender systems. Communications of the ACM, 40(3):56–58, 1997.

[10] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12):61–70, 1992.

[11] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In Recommender systems handbook, pages 1–35. 2011.

[12] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In The adaptive web, pages 325–341. Springer, 2007.

[13] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommendersystems: State of the art and trends. In Recommender systems handbook, pages 73–105. Springer, 2011.

[14] A. Zimdars, D. M. Chickering, and C. Meek. Using temporal data for making recom- mendations. In 17th Conference in Uncertainty in Artificial Intelligence, pages 580–588, 2001.

[15] Guy Shani, David Heckerman, and Ronen I Brafman. An mdp-based recommender system. Journal of Machine Learning Research, 6(Sep):1265–1295, 2005.

[16] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lil- licrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. arXiv preprint arXiv:1512.07679, 2015.

[17] Thorsten Joachims, Dayne Freitag, Tom Mitchell, et al. Webwatcher: A tour guide for the world wide web. In IJCAI (1), pages 770–777. Citeseer, 1997.

[18] Nima Taghipour, Ahmad Kardan, and Saeed Shiry Ghidary. Usage-based web recommen- dations: a reinforcement learning approach. In Proceedings of the 2007 ACM conference on Recommender systems, pages 113–120, 2007.

[19] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining. Communications of the ACM, 43(8):142–151, 2000.

[20] Nima Taghipour and Ahmad Kardan. A hybrid web recommender system based on q- learning. In Proceedings of the 2008 ACM symposium on Applied computing, pages 1164– 1168, 2008.

[21] Anongnart Srivihok and Pisit Sukonmanee. E-commerce intelligent agent: personalization travel support agent using q learning. In Proceedings of the 7th international conference on Electronic commerce, pages 287–292, 2005.

[22] Tariq Mahmood, Ghulam Mujtaba, and Adriano Venturini. Dynamic personalization in conversational recommender systems. Information Systems and e-Business Management, 12(2):213–238, 2014

[23] Binbin Hu, Chuan Shi, and Jian Liu. Playlist recommendation based on reinforcement learning. In International Conference on Intelligence Science, pages 172–182. Springer, 2017.

[24] Pornthep Rojanavasu, Phaitoon Srinil, and Ouen Pinngern. New recommendation system using reinforcement learning. Special Issue of the Intl. J. Computer, the Internet and Management, 13(SP 3), 2005.

[25] Mircea Preda and Dan Popescu. Personalized web recommendations: supporting epistemic information about end-users. In The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), pages 692–695. IEEE, 2005.

[26] Wacharawan Intayoad, Chayapol Kamyod, and Punnarumol Temdee. Reinforcement learning for online learning recommendation system. In 2018 Global Wireless Summit (GWS), pages 167–170. IEEE, 2018.

[27] Chung-Yi Chi, Richard Tzong-Han Tsai, Jeng-You Lai, and Jane Yung-jen Hsu. A re- inforcement learning approach to emotion-based automatic playlist generation. In 2010 International Conference on Technologies and Applications of Artificial Intelligence, pages 60–65. IEEE, 2010.

[28] Thorsten Bohnenberger and Anthony Jameson. When policies are better than plans: Decision-theoretic planning of recommendation sequences. In Proceedings of the 6th in- ternational conference on Intelligent user interfaces, pages 21–24, 2001.

[29] Tariq Mahmood and Francesco Ricci. Learning and adaptivity in interactive recommender systems. In Proceedings of the ninth international conference on Electronic commerce, pages 75–84, 2007.

[30] Tariq Mahmood and Francesco Ricci. Improving recommender systems with adaptive conversational strategies. In Proceedings of the 20th ACM conference on Hypertext and hypermedia, pages 73–82, 2009.

[31] Afsar, M. M., Crump, T., & Far, B. (2021). Reinforcement learning based recommender systems: A survey. *arXiv preprint arXiv:2101.06286*.

[32] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. Dj-mc: A reinforcement-learning agent for music playlist recommendation. arXiv preprint arXiv:1401.1880, 2014.

[33] Yu Wang. A hybrid recommendation for music based on reinforcement learning. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 91–103. Springer, 2020.

[34] Lixin Zou, Long Xia, Zhuoye Ding, Dawei Yin, Jiaxing Song, and Weidong Liu. Reinforcement learning to diversify top-n recommendation. In International Conference on Database Systems for Advanced Applications, pages 104–120. Springer, 2019.

[35] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In 2008 Eighth IEEE International Conference on Data Mining, pages 263–272. Ieee, 2008.

[36] Su Liu, Ye Chen, Hui Huang, Liang Xiao, and Xiaojun Hei. Towards smart educational recommendations with reinforcement learning in classroom. In 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), pages 1079–1084. IEEE, 2018.

[37] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, et al. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. arXiv preprint arXiv:1905.12767, 2019.

[38] Eleni Fotopoulou, Anastasios Zafeiropoulos, Michalis Feidakis, Dimitrios Metafas, and Symeon Papavassiliou. An interactive recommender system based on reinforcement learning for improving emotional competences in educational groups. In International Conference on Intelligent Tutoring Systems, pages 248–258. Springer, 2020.

[39] Floris Den Hengst, Mark Hoogendoorn, Frank Van Harmelen, and Joost Bosman. Reinforcement learning for personalized dialogue management. In IEEE/WIC/ACM International Conference on Web Intelligence, pages 59–67, 2019.

[40] Inigo Casanueva, Pawe l Budzianowski, Pei-Hao Su, Nikola Mrkšić, Tsung-Hsien Wen, Stefan Ultes, Lina Rojas-Barahona, Steve Young, and Milica Gašić. A benchmarking environment for reinforcement learning based task oriented dialogue management. arXiv preprint arXiv:1711.11023, 2017.

[41] Feng Liu, Huifeng Guo, Xutao Li, Ruiming Tang, Yunming Ye, and Xiuqiang He. End-to-end deep reinforcement learning based recommendation with supervised embedding. In Proceedings of the 13th International Conference on Web Search and Data Mining, pages 384–392, 2020.

[42] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... & Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

[43] Yufan Zhao, Donglin Zeng, Mark A Socinski, and Michael R Kosorok. Reinforcement learning strategies for clinical trials in nonsmall cell lung cancer. Biometrics, 67(4):1422–1433, 2011.

[44] Le, Q., & Mikolov, T. (2014, June). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188-1196). PMLR.

[45] https://flutter.dev/docs/resources/architectural-overview

[46] https://www.django-rest-framework.org

Résumé

Les systèmes de recommandation sont parmi les domaines les plus étudiés en intelligence artificielle à cause de l'émergence du Big data ainsi que leur importance dans une grande variété d'applications, telles que les achats en ligne, etc. De nombreuses approches ont été proposées afin de développer des systèmes de recommandation robustes qui correspondent aux intérêts des utilisateurs, tels que le filtrage collaboratif (FC), le filtrage basé sur le contenu (FBC) et les techniques hybrides. Plusieurs méthodes proposées considèrent l'interaction utilisateur-système comme un processus statique, ce qui dégrade la qualité des recommandations à long terme. Dans ce mémoire, nous avons proposé un modèle d'apprentissage par renforcement en ligne, appelé DeepTaste, pour la recommandation alimentaire. Le modèle proposé utilise Doc2vec pour l'extraction des caractéristiques d'état, et l'algorithme du k-plus proches voisins (kppv) pour générer la liste des éléments recommandés (TopN). L'étude expérimentale a montré que le modèle proposé peut détecter le changement des comportements des utilisateurs et fournir des recommandations adaptées aux intérêts des clients.

Mots clés : Apprentissage par renforcement, Systèmes de recommandation, DQN, Doc2vec, kppv.

Abstract

Recommendation systems are one of the most widely-studied domains in the field of artificialintelligence, due to the increasing of big data as well as their importance in wide variety of industriessuch as online shopping and entertainment industry and so on. A lot of approaches have been proposedto build robust recommendation systems that match users interests such as collaborative filtering (CF),content-based filtering (CBF) and hybrid techniques. Several proposed approaches take the user—system interaction as a static process which damage the user experience in the long run. In this thesiswe proposed an online model free reinforcement learning framework called DeepTaste for foodrecommendation. The proposed framework use Doc2vec for state features extraction and k-nearestneighbor (knn) for generating TopN-recommended items. It showed a quite interesting improvement ofcapturing users behaviors and delivering recommendations that fit users interests.

Keywords: Reinforcement learning, Recommender Systems, DQN, Doc2vec, knn.

ملخّص

تعد أنظمة التوصية واحدة من أكثر المجالات التي تمت دراستها على نطاق واسع في مجال الذكاء الاصطناعي ، نظرًا لزيادة البيانات الضخمة بالإضافة إلى أهميتها في مجموعة متنوعة من الصناعات مثل التسوق عبر الإنترنت وصناعة الترفيه وما إلى ذلك. تم اقتراح الكثير من الأساليب لبناء أنظمة توصية قوية تتوافق مع اهتمامات المستخدمين مثل التصفية التعاونية (CF) ، والتصفية القائمة على المحتوى (CBF) والتقنيات الهجينة. العديد من الأساليب المقترحة تأخذ تفاعل المستخدم مع النظام كعملية ثابتة تضر بتجربة المستخدم على المدى الطويل. في هذه الأطروحة اقترحنا نموذجًا مجانيًا لإطار عمل التعلم المعزز عبر الإنترنت يسمى DeepTaste للتوصية الغذائية. يستخدم إطار العمل المقترح Doc2vec لاستخراج ميزات الحالة و k-الجار الأقرب (knn) لتوليد عناصر TopN الموصى بها. لقد أظهر تحسنًا مثيرًا للاهتمام في التقاط سلوكيات المستخدمين وتقديم التوصيات التي تناسب اهتمامات المستخدمين.

الكلمات المفتاحية: التعلم المعزز ، أنظمة التوصية ، DQN، Doc2vec، knn