|  |  |  |
| --- | --- | --- |
|  | UNIVERSITY OF CRAIOVA  FACULTY OF AUTOMATION, COMPUTERS AND ELECTRONICS  COMPUTERS AND INFORMATION TECHNOLOGY DEPARTMENT |  |

DEGREE PROJECT

Mahmoud Rasmi Vlad

SCIENTIFIC COORDINATOR

Prof.univ.dr. ing. Stănescu Liana

JULY 2017

CRAIOVA

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITY OF CRAIOVA  FACULTY OF AUTOMATION, COMPUTERS AND ELECTRONICS  COMPUTERS AND INFORMATION TECHNOLOGY DEPARTMENT |  |

“Application for Patient Management in a Medical Clinic”

Mahmoud Rasmi Vlad

SCIENTIFIC COORDINATOR

Prof.univ.dr. ing. Stănescu Liana

JULY 2017

CRAIOVA

*„Învățătura este o comoară care își urmează stăpânul pretutindeni.”*

Proverb popular

**DECLARAȚIE DE ORIGINALITATE**

Subsemnatul **Mahmoud Rasmi Vlad**, student la specializarea Calculatoare în Limba Engleză din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoştinţă de cele prezentate mai jos şi că îmi asum, în acest context, originalitatea proiectului meu de licenţă:

* cu titlul “Application for Patient Management in a Medical Clinic”
* coordonată de Prof.univ.dr. ing. Stănescu Liana,
* prezentată în sesiunea IULIE 2017.

La elaborarea proiectului de licenţă, se consideră plagiat una dintre următoarele acţiuni:

* reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele şi referinţa precisă,
* redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
* prezentarea unor date experimentale obţinute sau a unor aplicaţii realizate de alţi autori fără menţionarea corectă a acestor surse,
* însuşirea totală sau parţială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situaţii neplăcute se recomandă:

* plasarea între ghilimele a citatelor directe şi indicarea referinţei într-o listă corespunzătoare la sfărşitul lucrării,
* indicarea în text a reformulării unei idei, opinii sau teorii şi corespunzător în lista de referinţe a sursei originale de la care s-a făcut preluarea,
* precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele et caetera,
* precizarea referinţelor poate fi omisă dacă se folosesc informaţii sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data, Semnătura candidatului,

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  Facultatea de Automatică, Calculatoare şi Electronică  Departamentul de Calculatoare și Tehnologia Informației | Aprobat la data de  …………………  Şef de departament,  Prof. dr. ing.  Marius BREZOVAN |

**PROIECTUL DE DIPLOMĂ**

|  |  |
| --- | --- |
| Numele și prenumele studentului/-ei: | Mahmoud Rasmi Vlad |
| Enunțul temei: | “Application for Patient Management in a Medical Clinic” este o aplicație web pentru o buna organizare in cadrul unei clinici medicale. |
| Datele de pornire: | Studiul Python; Django Framework; Django JET ; API-uri pentru trimiterea emailurilor si a SMS-urilor |
| Conținutul proiectului: | Introduction – Application concepts, Purpose, Motivation.  Technologies and frameworks - Theoretical aspects.  System Design and Implementation of the project. |
| Material grafic obligatoriu: |  |
| Consultații: | Periodice |
| Conducătorul științific  (titlul, nume și prenume, semnătura): | Prof.univ.dr. ing. Stănescu Liana |
| Data eliberării temei: |  |
| Termenul estimat de predare a proiectului: |  |
| Data predării proiectului de către student și semnătura acestuia: |  |

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  Facultatea de Automatică, Calculatoare şi Electronică  Departamentul de [Calculatoare și Tehnologia Informației / Automatică și Electronică / Mecatronică și Robotică] |  |

**REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC**

|  |  |
| --- | --- |
| Numele și prenumele candidatului/-ei: |  |
| Specializarea: | [*Denumirea oficială a specializării absolvite de candidat*] |
| Titlul proiectului: | [*Titlul lucrării*] |
| Locația în care s-a realizat practica de documentare (se bifează una sau mai multe din opțiunile din dreapta): | În facultate □ |
| În producție □ |
| În cercetare □ |
| Altă locație: [*se detaliază*] |

În urma analizei lucrării candidatului au fost constatate următoarele:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nivelul documentării | | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Tipul proiectului | | Cercetare  □ | Proiectare  □ | Realizare practică □ | Altul  [*se detaliază*] |
| Aparatul matematic utilizat | | Simplu  □ | Mediu  □ | Complex □ | Absent  □ |
| Utilitate | | Contract de cercetare □ | Cercetare internă □ | Utilare  □ | Altul  [*se detaliază*] |
| Redactarea lucrării | | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Partea grafică, desene | | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Realizarea practică | Contribuția autorului | Insuficientă  □ | Satisfăcătoare □ | Mare  □ | Foarte mare  □ |
| Complexitatea  temei | Simplă  □ | Medie  □ | Mare  □ | Complexă  □ |
| Analiza cerințelor | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Arhitectura | Simplă  □ | Medie  □ | Mare  □ | Complexă  □ |
| Întocmirea specificațiilor funcționale | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Implementarea | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Testarea | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Funcționarea | Da  □ | Parțială  □ | Nu  □ | |
| Rezultate experimentale | | Experiment propriu  □ | | Preluare din bibliografie  □ | |
| Bibliografie | | Cărți | Reviste | Articole | Referințe web |
| Comentarii  și  observații | |  | | | |

În concluzie, se propune:

|  |  |
| --- | --- |
| ADMITEREA PROIECTULUI  □ | RESPINGEREA PROIECTULUI  □ |

Data, Semnătura conducătorului științific,

**PROJECT SUMMARY**

Application for Patient Management in a Medical Clinic” is a web application designed for Medical Clinics (with three levels of access: Admin, Nurse and Doctor.). The application offers a good management of the entire medical process.

The appointment process is now simplified and also staff from the reception, nurses or doctors can set an appointment for a patient at a specialization and at a specific doctor. When a new appointment is made for a doctor, the doctor will receive a text message with the date and hour of the new appointment.

The doctor has the possibility to choose the diagnostic for a consultation, to fill the consultation results which will be send to the patient via email or can be pick up from the clinic the next day or after receiving an email from the clinic.

For the development of the application was used Python programming language.

As a tools the following were used:

* PyCharm (python based IDE)
* Django framework
* Django JET
* FusionCharts
* Twilio (as an API for sending text messages)

The architecture of the application follows the Model-View-Template (MVT) is slightly different from MVC. In fact the main difference between the two patterns is that Django itself handles the Controller part (Code which controls the interactions between the Model and the View), leaving the programmers with the template. The template is mixed between HTML and Django Template Language (DTL).

***Key Words***: Django, Django-JET, Python

**Acknowledgements**

I would like to thank you specially my coordinator MRS. Liana STANESCCU who was a real support during whole period of development. I offer to her my sincere thanks, who was a great help with advice.

In addition I would like to thank whole department of COMPUTERS AND INFORMATION TECHNOLOGY from Faculty of Automation, Computers and Electronics for all the valuable lessons that I have learn related to this field and not only.

Not in the last, I would like to thank you my family, my friends which gave me an enormous support during though moments in these 4 years.

**PROLOGUE**

Do you think is time for a change?

Are you tired of configuration programs? Do you want to just use them and work properly?

Programs are a bit too complex for your needs? Try MedApp a simple patient management system for a clinic.

No pre-configuration needed, not too complex, simple design and awesome functionality.

Contents

[1 INTRODUCTION 1](#_Toc486719156)

[2 Technologies and frameworks 2](#_Toc486719157)

[2.1 Python (programming language) 2](#_Toc486719158)

[2.2 Django 7](#_Toc486719159)

[(The web framework for perfectionists with deadlines) 7](#_Toc486719160)

[2.4 Django JET 14](#_Toc486719161)

[2.5 Twilio 15](#_Toc486719162)

[2.6 PyCharm 16](#_Toc486719163)

[Debugging the Application 17](#_Toc486719164)

[Code Refactoring 18](#_Toc486719165)

[3 SYSTEM DESIGN AND IMPLEMENTATION 19](#_Toc486719166)

[3.1 System requirements 19](#_Toc486719167)

[3.2 Use case diagram 22](#_Toc486719168)

[3.3 System architecture 23](#_Toc486719169)

[3.4 Database design 24](#_Toc486719170)

[3.5 Project structure 26](#_Toc486719171)

[3.5.1 Class Diagram Admin.py 27](#_Toc486719172)

[3.5.2 Class diagram Models.py 28](#_Toc486719173)

[3.6 Implementation 29](#_Toc486719174)

[3.6.1 Sending e-mail when new results are added 29](#_Toc486719175)

[3.6.2 Sending text message when new appointment is added 30](#_Toc486719176)

[3.6.3 Filter for viewing only relevant information based on logged in user 31](#_Toc486719177)

[3.6.4 Charts 33](#_Toc486719178)

[3.6.5 Django Password management 34](#_Toc486719179)

**LISTA FIGURILOR**

[Figura 1. Selectarea prin click dreapta a opțiunii „Update field” **Error! Bookmark not defined.**](#_Toc309893908)

[Figura 2. Actualizarea întregului tabel **Error! Bookmark not defined.**](#_Toc309893909)

**LISTA TABELELOR**

[Tabelul 1. Nume de utilizatori și valorile rezumat ale parolelor acestora **Error! Bookmark not defined.**](#_Toc309893145)

`

# 1 INTRODUCTION

#### Purpose

The main purpose of “Application for Patient Management in a Medical Clinic” is to simplify the entire medical process from a Medical Clinic. In order to avoid the long queues from the clinics, just to make an appointment, the patient can just call and set it. The patient will receive also a text message with the confirmation and the details of the appointments, furthermore also the doctors will receive a text message with the details of the new appointment.

The application is designed to be user friendly and simple to use.

After the consultation, the doctor fill a form with the results. Is not needed that the patient to go for results because they will be send via e-mail.

#### Motivation

One of the reasons I’ve chosen to develop such an application came from my personal experiences I had with existing products on our market. Despite the fact, that our market is full with almost similar products every one of them had issues that can generate waste of time. Some of them needs a hard configuration in order to start working, other loose the connection with their database (in a simple way of speaking they stop working), or their User-Interface is full of unimportant things making them hard to use. Those are problems that could not be accepted in the medical process, where time is the most important factor.

The medical system needs well-structured applications in order to simplify the entire process. The processes like patient-registration, patient-consultation and patient-results are the main time-consuming. Providing solution for those problems generates a visible comfort for the medical staff and for the patients.

I’ve taken this as challenge and an opportunity of self-improvement. I’ve tried to develop an application that avoids the problems I’ve experienced, but I am aware this is not a perfect product and further major improvements could be done. For me it was worth it because I’ve had the chance to discover new technologies and to design it how I would like to look and work.

# Technologies and frameworks



## 2.1 Python (programming language)

“Over six years ago, in December 1989, I was looking for a "hobby" programming project

that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus). ”- about the origin of Python, Van Rossum 1996

Python is a widely used high-level programming language for general-purpose programming.

An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

Python is a multi-paradigm programming language: object-oriented programming and

structured programming are fully supported, and many language features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

The design of Python offers some support for functional programming in the Lisp tradition.

The language has map(), reduce() and filter() functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

Language can be summarized with a few aphorisms:

* Beautiful is better than ugly
* Explicit is better than explicit
* Simple is better than complex
* Readability counts

Python is intended to be a highly readable language. It is designed to have an uncluttered visual layout, often using English keywords where other languages use punctuation. Python doesn't have semicolons and curly brackets "{}" which is different compared to most of the programming language. Further, Python has fewer syntactic exceptions and special cases than [C](https://en.wikipedia.org/wiki/C_%28programming_language%29) or [Pascal](https://en.wikipedia.org/wiki/Pascal_%28programming_language%29).

Python uses [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [blocks](https://en.wikipedia.org/wiki/Block_%28programming%29) – rather than [curly braces](https://en.wikipedia.org/wiki/Curly_bracket_programming_language) or keywords. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. This feature is also sometimes termed the [off-side rule](https://en.wikipedia.org/wiki/Off-side_rule). (Wikipedia, 2010).

Python is a general-purpose, versatile and popular programming language. It's great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer's stack as it can be used for everything from web development to software development and scientific applications.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective. (Python Software Foundation, 2001-2007)

Python is often compared to other interpreted languages such as Java, JavaScript, Perl, Tcl, or Smalltalk. Comparisons to C++, Common Lisp and Scheme can also be enlightening. In this section I will briefly compare Python to each of these languages. These comparisons concentrate on language issues only. In practice, the choice of a programming language is often dictated by other real-world constraints such as cost, availability, training, and prior investment, or even emotional attachment. Since these aspects are highly variable, it seems a waste of time to consider them much for this comparison.

Python programs are generally expected to run slower than Java programs, but they also take much less time to develop. Python programs are typically 3-5 times shorter than equivalent Java programs. This difference can be attributed to Python's built-in high-level data types and its dynamic typing. For example, a Python programmer wastes no time declaring the types of arguments or variables, and Python's powerful polymorphic list and dictionary types, for which rich syntactic support is built straight into the language, find a use in almost every Python program. Because of the run-time typing, Python's run time must work harder than Java's. For example, when evaluating the expression a+b, it must first inspect the objects a and b to find out their type, which is not known at compile time. It then invokes the appropriate addition operation, which may be an overloaded user-defined method. Java, on the other hand, can perform an efficient integer or floating point addition, but requires variable declarations for a and b, and does not allow overloading of the + operator for instances of user-defined classes.

For these reasons, Python is much better suited as a "glue" language, while Java is better characterized as a low-level implementation language. In fact, the two together make an excellent combination. Components can be developed in Java and combined to form applications in Python; Python can also be used to prototype components until their design can be "hardened" in a Java implementation. To support this type of development, a Python implementation written in Java is under development, which allows calling Python code from Java and vice versa. In this implementation, Python source code is translated to Java bytecode (with help from a run-time library to support Python's dynamic semantics).

Python's "object-based" subset is roughly equivalent to JavaScript. Like JavaScript (and unlike Java), Python supports a programming style that uses simple functions and variables without engaging in class definitions. However, for JavaScript, that's all there is. Python, on the other hand, supports writing much larger programs and better code reuse through a true object-oriented programming style, where classes and inheritance play an important role.

Python and Perl come from a similar background (Unix scripting, which both have long outgrown), and sport many similar features, but have a different philosophy. Perl emphasizes support for common application-oriented tasks, e.g. by having built-in regular expressions, file scanning and report generating features. Python emphasizes support for common programming methodologies such as data structure design and object-oriented programming, and encourages programmers to write readable (and thus maintainable) code by providing an elegant but not overly cryptic notation. As a consequence, Python comes close to Perl but rarely beats it in its original application domain; however Python has an applicability well beyond Perl's niche.

Like Python, Tcl is usable as an application extension language, as well as a stand-alone programming language. However, Tcl, which traditionally stores all data as strings, is weak on data structures, and executes typical code much slower than Python. Tcl also lacks features needed for writing large programs, such as modular namespaces. Thus, while a "typical" large application using Tcl usually contains Tcl extensions written in C or C++ that are specific to that application, an equivalent Python application can often be written in "pure Python". Of course, pure Python development is much quicker than having to write and debug a C or C++ component. It has been said that Tcl's one redeeming quality is the Tk toolkit. Python has adopted an interface to Tk as its standard GUI component library.

Tcl 8.0 addresses the speed issuse by providing a bytecode compiler with limited data type support, and adds namespaces. However, it is still a much more cumbersome programming language.

Perhaps the biggest difference between Python and Smalltalk is Python's more "mainstream" syntax, which gives it a leg up on programmer training. Like Smalltalk, Python has dynamic typing and binding, and everything in Python is an object. However, Python distinguishes built-in object types from user-defined classes, and currently doesn't allow inheritance from built-in types. Smalltalk's standard library of collection data types is more refined, while Python's library has more facilities for dealing with Internet and WWW realities such as email, HTML and FTP.

Python has a different philosophy regarding the development environment and distribution of code. Where Smalltalk traditionally has a monolithic "system image" which comprises both the environment and the user's program, Python stores both standard modules and user modules in individual files which can easily be rearranged or distributed outside the system. One consequence is that there is more than one option for attaching a Graphical User Interface (GUI) to a Python program, since the GUI is not built into the system.

Almost everything said for Java also applies for C++, just more so: where Python code is typically 3-5 times shorter than equivalent Java code, it is often 5-10 times shorter than equivalent C++ code! Anecdotal evidence suggests that one Python programmer can finish in two months what two C++ programmers can't complete in a year. Python shines as a glue language, used to combine components written in C++.

## 2.2 Django



## (The web framework for perfectionists with deadlines)

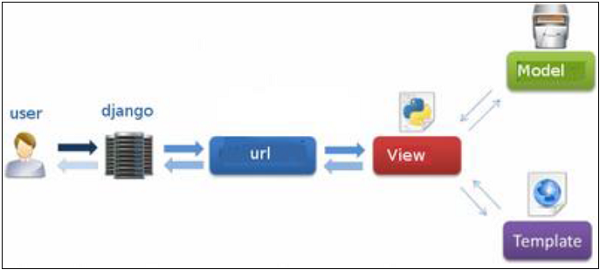
Django grew organically from real-world applications written by a Web development team in Lawrence, Kansas, USA. It was born in the fall of 2003, when the Web programmers at the Lawrence Journal-World newspaper, Adrian Holovaty and Simon Willison, began using Python to build applications.

The World Online team, responsible for the production and maintenance of several local news sites, thrived in a development environment dictated by journalism deadlines. For the sites – including LJWorld.com, Lawrence.com and KUsports.com – journalists (and management) demanded that features be added and entire applications be built on an intensely fast schedule, often with only days’ or hours’ notice. Thus, Simon and Adrian developed a time-saving Web development framework out of necessity – it was the only way they could build maintainable applications under the extreme deadlines.

In summer 2005, after having developed this framework to a point where it was efficiently powering most of World Online’s sites, the team, which now included Jacob Kaplan-Moss, decided to release the framework as open source software. They released it in July 2005 and named it Django, after the jazz guitarist Django Reinhardt.

Django lets you build deep, dynamic, interesting sites in an extremely short time. Django is designed to let you focus on the fun, interesting parts of your job while easing the pain of the repetitive bits. In doing so, it provides high-level abstractions of common Web development patterns, shortcuts for frequent programming tasks, and clear conventions on how to solve problems. At the same time, Django tries to stay out of your way, letting you work outside the scope of the framework as needed. (TheDjangoBook, 2016)

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.



Django is a free and open-source web framework, written in Python, which follows the model-view-template (MVT) architectural pattern. As I already said in the introduction, Django stays under the Mode-View-Template architectural pattern which is nearly the same with the MVC pattern beside the part with the Controller logic which is taken under Django task, leaving the user to focus over the templates.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "plug ability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Some well-known sites that use Django include the Public Broadcasting Service, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket, and Nextdoor. Included in Django framework core are a set of useful task such as: a lightweight and standalone web server for development and testing; a lightweight and standalone web server for development and testing; an internationalization system, including translations of Django's own components into a variety of languages.

Django has embedded in its distribution a number of applications, those can be found in Django “contrib” packages that users can only use with needing to write them again such as: an extensible authentication system; tools for generating Google Sitemaps; framework for creating GIS applications.

Although all its embedded feature Django system allows third party code to be plugged in regular projects, following the reusable apps conventions. Now are more than 2500 package available ready to extend the framework core functionality providing solutions for problems that in origin Django cannot handle (registration, search, API provision and consumption, CMS).

Django supports several major database engines and you can set up any of them based on your comfort: MySQL; PostgreSQL; SQLite 3; Oracle; Mongo DB. Django comes with a web server for developing and testing applications. This server is pre-configured to work with Django, and more importantly, it restarts whenever you modify the code.

Now let’s focus on Django advantages, which not only acts as a quick solution for web development, its ability to deliver high-quality code and transparent writing provides the right platform to customers for business and even for the developers.

* Fast: This has been designed in a way to help the developers make an application as fast as possible. From idea, production to release, Django helps in making it both cost effective and efficient. Thus it becomes an ideal solution for developers having a primary focus on deadlines.
* Fully Loaded: It works in a way that includes dozens of extras to help with user authentication, site maps, content administration, RSS feeds and much more such things. These aspects help in carrying out the web development process completely.
* Secure: When you are doing it in Django, it is ensured that developers don’t commit any mistakes related to security. Some of the common mistakes include SQL injection, cross-site request forgery, clickjacking and cross-site scripting. To manage effectively usernames and passwords, the user authentication system is the key.
* Scalable: To meet the heaviest traffic demand, the benefits of Django framework can be seen. Therefore, the busiest sites use this medium to quickly meet the traffic demands.
* Versatile: Content management, scientific computing platforms, and even big organizations, all these aspects are very efficiently managed by the use of Django.

Probably PHP is the toughest competitor, and both are immensely popular. But there are many advantages of Django over PHP, which makes the former a favorite choice among the developers. Deciding the best one is not that simple, but these comparisons will help you in doing so.

* Better Design: While PHP is specifically designed for web designs, Django bases on more robust language. So, to write a good code, it is easier to do it in python compared to PHP.
* Better Web Framework: The market is fragmented owing to so many great frameworks. But while frameworks have an upfront cost, it should have a long-term value. And here, Django is a clear winner.
* Readability Factor: PHP follows a classic approach where Python has more strict identification enforcements. Hence the best.
* Simpler Syntax: The approach is much simpler and the codes are easy when doing it in Python.
* Debugging Tools: The tools that come with Python package are sufficient enough to handle issues.
* Package Management: It is good in Python by which one can write, read and share packages through which other developers can easily use it in other applications.

Along with the advantages, come the disadvantages. There are many disadvantages of Django mentioned below:

* Uses routing pattern specify its URL
* Django is too monolithic
* Everything is based on Django ORM
* Components get deployed together
* Knowledge of full system is required to work.

Although there are many pros and cons of Django, however, when a project with a deadline is considered, using Django for the project provides the ultimate solution. And to speed it up even more, the custom user model configuration cab be installed. Earlier, when developing a code requires a lot of time; with its help, it became far easier. The development cost is reduced to almost 80% and outsourcing is completely shattered.



**2.3 SQLite**

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. Think of SQLite not as a replacement for Oracle but as a replacement for fopen().

SQLite is a compact library. With all features enabled, the library size can be less than 500KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function inlining and loop unrolling can cause the object code to be much larger.) If optional features are omitted, the size of the SQLite library can be reduced below 300KiB. SQLite can also be made to run in minimal stack space (4KiB) and very little heap (100KiB), making SQLite a popular database engine choice on memory constrained gadgets such as cellphones, PDAs, and MP3 players. There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments.

SQLite is very carefully tested prior to every release and has a reputation for being very reliable. Most of the SQLite source code is devoted purely to testing and verification. An automated test suite runs millions and millions of test cases involving hundreds of millions of individual SQL statements and achieves 100% branch test coverage. SQLite responds gracefully to memory allocation failures and disk I/O errors. Transactions are ACID even if interrupted by system crashes or power failures. All of this is verified by the automated tests using special test harnesses which simulate system failures. Of course, even with all this testing, there are still bugs. But unlike some similar projects (especially commercial competitors) SQLite is open and honest about all bugs and provides bugs lists and minute-by-minute chronologies of code changes.

The SQLite code base is supported by an international team of developers who work on SQLite full-time. The developers continue to expand the capabilities of SQLite and enhance its reliability and performance while maintaining backwards compatibility with the published interface spec, SQL syntax, and database file format. The source code is absolutely free to anybody who wants it, but professional support is also available.

The SQLite project was started on 2000-05-09. The future is always hard to predict, but the intent of the developers is to support SQLite through the year 2050. Design decisions are made with that objective in mind.

We the developers hope that you find SQLite useful and we entreat you to use it well: to make good and beautiful products that are fast, reliable, and simple to use. Seek forgiveness for yourself as you forgive others. And just as you have received SQLite for free, so also freely give, paying the debt forward.

SQLite is not directly comparable to client/server SQL database engines such as MySQL, Oracle, PostgreSQL, or SQL Server since SQLite is trying to solve a different problem.

Client/server SQL database engines strive to implement a shared repository of enterprise data. They emphasis scalability, concurrency, centralization, and control. SQLite strives to provide local data storage for individual applications and devices. SQLite emphasizes economy, efficiency, reliability, independence, and simplicity. SQLite does not compete with client/server databases. SQLite competes with fopen().

SQLite works great as the database engine for most low to medium traffic websites (which is to say, most websites). The amount of web traffic that SQLite can handle depends on how heavily the website uses its database. Generally speaking, any site that gets fewer than 100K hits/day should work fine with SQLite. The 100K hits/day figure is a conservative estimate, not a hard upper bound. SQLite has been demonstrated to work with 10 times that amount of traffic.

The SQLite website uses SQLite itself, of course, and as of this writing (2015) it handles about 400K to 500K HTTP requests per day, about 15-20% of which are dynamic pages touching the database. Dynamic content uses about 200 SQL statements per webpage. This setup runs on a single VM that shares a physical server with 23 others and yet still keeps the load average below 0.1 most of the time.

SQLite does not need to be "installed" before it is used. There is no "setup" procedure. There is no server process that needs to be started, stopped, or configured. There is no need for an administrator to create a new database instance or assign access permissions to users. SQLite uses no configuration files. Nothing needs to be done to tell the system that SQLite is running. No actions are required to recover after a system crash or power failure. There is nothing to troubleshoot. SQLite just works.

Most SQL database engines are implemented as a separate server process. Programs that want to access the database communicate with the server using some kind of interprocess communication (typically TCP/IP) to send requests to the server and to receive back results. SQLite does not work this way. With SQLite, the process that wants to access the database reads and writes directly from the database files on disk. There is no intermediary server process.

There are advantages and disadvantages to being server less. The main advantage is that there is no separate server process to install, setup, configure, initialize, manage, and troubleshoot. This is one reason why SQLite is a "zero-configuration" database engine. Programs that use SQLite require no administrative support for setting up the database engine before they are run. Any program that is able to access the disk is able to use a SQLite database.

On the other hand, a database engine that uses a server can provide better protection from bugs in the client application - stray pointers in a client cannot corrupt memory on the server. And because a server is a single persistent process, it is able control database access with more precision, allowing for finer grain locking and better concurrency.

Most SQL database engines are client/server based. Of those that are server less, SQLite is the only one that this author knows of that allows multiple applications to access the same database at the same time.

When optimized for size, the whole SQLite library with everything enabled is less than 500KiB in size (as measured on an ix86 using the "size" utility from the GNU compiler suite.) Unneeded features can be disabled at compile-time to further reduce the size of the library to under 300KiB if desired.

Most other SQL database engines are much larger than this. IBM boasts that its recently released CloudScape database engine is "only" a 2MiB jar file - an order of magnitude larger than SQLite even after it is compressed! Firebird boasts that its client-side library is only 350KiB. That's as big as SQLite and does not even contain the database engine. The Berkeley DB library from Oracle is 450KiB and it omits SQL support, providing the programmer with only simple key/value pairs.

## 2.4 Django JET



Jet is a modern template for Django with improved functionality. Jet is a desktop and mobile user experience combine in one single package.

Features:

* New fresh look

JET brings a breath of fresh air to the world of Django. Your super complex systems will be cool not only inside, but outside too, and you won't need to feel a shame anymore showing admin interface to your clients

* Responsive mobile interface

Administration interface specially optimized for mobile devices requiring no action from your side.

* Useful home page

Using your home page just as a bunch of links to other pages? Transform it to a complete dashboard with widgets that will give you a quick access to useful tools and info. You can use built-in widgets or create your own.

* Minimal template overriding

We've worked hard to keep original Django layout and make JET compatible with 3rd party packages – only 3 Django admin base templates (base.html, popup\_response.html and includes/fieldset.html) are overridden without any significant changes.

* Easy integration with your system

Developing your Django-based system for a long time, but want to make it look fresh without rewriting even a single line of code? You can just install JET and get the same functionality with a completely different look

* Auto completion

You can transform all your foreign key fields and filters with large amount of data to AJAX-based dropdowns with just a few lines of code

## 2.5 Twilio

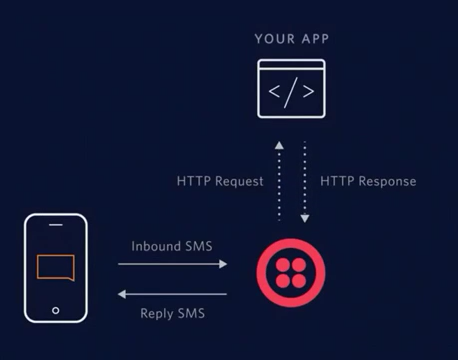


Twilio is a cloud communications platform with build in blocks for messaging, voice and video that could be easily integrate in your web and mobile applications. It is taken care of the infrastructure so you don’t have to. This is really helpful and time reducing feature, now you can focus only on your app in the programming language you already use.

Twilio uses Amazon Web Services to host telephony infrastructure and provide connectivity between HTTP and the public switched telephone network (PSTN) through its APIs. It follows a set of architectural design principles to protect against unexpected outages.

Voice over IP (VoIP) is a technology that involves sending telephone calls over data networks, such as the Internet, rather than the traditional Public Switched Telephone Network ‘PSTN’. Traditional phone calls across the PSTN use a dedicated circuit that transfers calls as uncut streams, allowing no other information on the circuit regardless of available bandwidth. Conversely, IP networks transfer data more efficiently in packets that get reassembled on the receiving end rather than using a dedicated circuit. For a VoIP call, audio is first converted from an analog signal to a digital signal through a codec, separated into discrete packets, sent across an IP-based network, put back into order at the termination point, and then converted back to an analog signal to create audio that the end user can recognize. There are three major VoIP protocol standards—namely, H.323, SIP, and MGCP.

The Twilio REST API is a part of a scalable cloud communications infrastructure that handles all the complicated telephony required to send/receive calls and text messages so that you don't have to worry about anything beyond GET, POST and XML, what you're used to working with every day as a web developer.



Sending an SMS or MMS is one of the most common tasks performed on the Twilio Platform. Sending a message is as simple as POST-ing to the Messages resource, but since it’s a common action it’s worth walking through in detail below.

## 2.6 PyCharm



PyCharm is an [Integrated Development Environment](https://en.wikipedia.org/wiki/Integrated_Development_Environment) (IDE) used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming), specifically for the [Python](https://en.wikipedia.org/wiki/Python_%28programming_language%29) language. It is developed by the Czech company [JetBrains](https://en.wikipedia.org/wiki/JetBrains).[[2]](https://en.wikipedia.org/wiki/PyCharm#cite_note-2) It provides code analysis, a graphical debugger, an integrated unit tester, integration with [version control systems](https://en.wikipedia.org/wiki/Revision_control) (VCSes), and supports web development with [Django](https://en.wikipedia.org/wiki/Django_%28web_framework%29).

Features:

* PyCharm is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), with Windows, [macOS](https://en.wikipedia.org/wiki/MacOS) and Linux versions
* Project and Code Navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages
* Coding Assistance and [Analysis](https://en.wikipedia.org/wiki/Code_analysis), with [code completion](https://en.wikipedia.org/wiki/Autocomplete), syntax and error highlighting, linter integration, and quick fixes
* Python [Refactoring](https://en.wikipedia.org/wiki/Refactoring): including rename, extract method, introduce variable, introduce constant, pull up, push down and others
* Support for web frameworks: [Django](https://en.wikipedia.org/wiki/Django_%28web_framework%29), [web2py](https://en.wikipedia.org/wiki/Web2py) and [Flask](https://en.wikipedia.org/wiki/Flask_%28web_framework%29)
* Integrated Python [Debugger](https://en.wikipedia.org/wiki/Debugger)
* Integrated [Unit Testing](https://en.wikipedia.org/wiki/Unit_Testing), with line-by-line [code coverage](https://en.wikipedia.org/wiki/Code_coverage)
* [Google App Engine](https://en.wikipedia.org/wiki/Google_App_Engine) Python Development
* Version Control Integration: unified user interface for [Mercurial](https://en.wikipedia.org/wiki/Mercurial), [Git](https://en.wikipedia.org/wiki/Git_%28software%29), [Subversion](https://en.wikipedia.org/wiki/Apache_Subversion), [Perforce](https://en.wikipedia.org/wiki/Perforce) and [CVS](https://en.wikipedia.org/wiki/Concurrent_Versions_System) with changelists and merge

The PyCharm IDE starts with a menu at the top and underneath this, you have tabs for each open project.On the right side of the screen are debugging options for stepping through code.The left pane has a list of project files and external libraries.

To add a file you right-click on the project name and choose "new". You then get the option to add one of the following file types:

* File
* Directory
* Python Package
* Python File
* Jupyter Notebook
* HTML File
* Stylesheet
* JavaScript
* TypeScript
* CoffeeScript
* Gherkin
* Data Source

When you add a file, such as a python file you can start typing into the editor in the right panel.

The text is all color coded and has bold text. A vertical line shows the indentation so you can be sure that you are tabbing correctly.

The editor also includes full IntelliSense, which means as you start typing the names of libraries or recognized commands you can complete the commands by pressing tab.

### Debugging the Application

You can debug your application at any point by using the debugging options in the top right corner. If you are developing a graphical application then you can simply press the green button to run the application. You can also press shift and F10.

To debug the application you can either click the button next to the green arrow or press shift and F9.You can place breakpoints in the code so that the program stops on a given line by clicking in the gray margin on the line you wish to break at.

To make a single step forward you can press F8 which steps over the code. This means it will run the code but it won't step into a function. To step into the function you would press F7. If you are in a function and want to step out to the calling function press shift and F8.

At the bottom of the screen whilst you are debugging you will see various windows such as a list of processes and threads, and variables that you are watching the values for.

As you are stepping through the code you can add a watch to a variable so that you can see when the value changes.

Another great option is to run the code with coverage checker. The programming world has changed a lot over the years and now it is common for developers to perform test-driven development so that every change they make they can check to make sure they haven't broken another part of the system.

The coverage checker actually helps you to run the program, perform some tests and then when you have finished it will tell you how much of the code was covered as a percentage during your test run.

There is also a tool for showing the name of a method or class, how many times the items were called, and how long was spent in that particular piece of code.

### Code Refactoring

A really powerful feature of PyCharm is the code refactoring option.

When you start to develop code little marks will appear in the right margin. If you type something which is likely to cause an error or just isn't written well then PyCharm will place a colored marker.

Clicking on the colored marker will tell you the issue and will offer a solution.

For example, if you have an import statement which imports a library and then don't use anything from that library not only will the code turn gray the marker will state that the library is unused.

Other errors that will appear are for good coding, such as only having one blank line between an import statement and the start of a function. You will also be told when you have created a function that isn't in lowercase.

You don't have to abide by all of the PyCharm rules. Many of them are just good coding guidelines and are nothing to do with whether the code will run or not.

The code menu has other refactoring options. For example, you can perform code cleanup and you can inspect a file or project for issues.

.

# SYSTEM DESIGN AND IMPLEMENTATION

“Application for Patient Management in a Medical Clinic” is a web application designed for Medical Clinics (with three levels of access: Admin, Nurse and Doctor.). The application offers a good management of the entire medical process.

## 3.1 System requirements

“Application for Patient Management in a Medical Clinic” has the following functional requirements based on the three access level:

One access level is the application ADMIN who can control the following items:

* Possibility to add new access level for users:
* Add a new group name
* Specify the permissions for the new group
* Possibility to add the new users:
* Register a new user with e-mail and password
* Add the user in one access level.
* Change the existing password for one user
* Possibility to add a new specialty:
* Register a new specialty to the app
* Add an existing user or add a new user to

Another access level is NURSE who can control the following items:

* Patient management:
* Register a new patient with first name, last-name, address, CNP, date of birth, email or telephone number
* View existing patients
* Modify existing patients
* Delete specific patient
* Appointments management:
* Add a new appointment with time, doctor, patient and specialization
* View existing appointments
* Modify existing appointment
* Delete specific appointment
* Consultation management:
* Add new consultation with doctor, Specialization and patient
* View existing consultations
* Modify existing consultation
* Delete specific consultation

Other access level is DOCTOR who can control the following items:

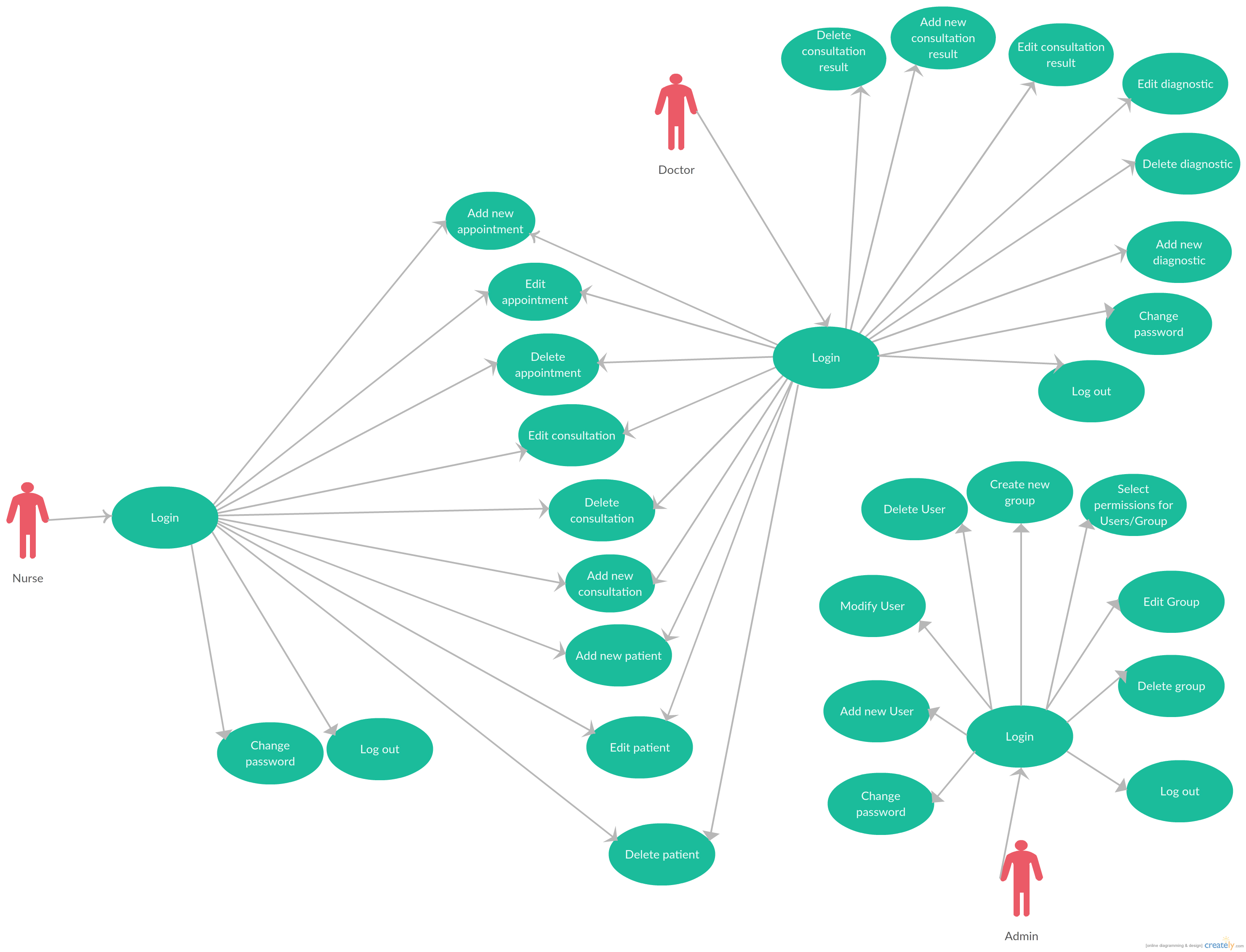
* Patient management with restriction for his patients
* Register a new patient with first name, last-name, address, CNP, date of birth, email or telephone number
* View existing patients
* Modify existing patients
* Delete specific patient
* Appointments management with restriction for his appointments
* Add a new appointment with time, doctor, patient and specialization
* View existing appointments
* Modify existing appointment
* Delete specific appointment
* Consultation management with restriction for his consultation
* Add new consultation with doctor, Specialization and patient
* View existing consultations
* Modify existing consultation
* Delete specific consultation
* Consultation result management with restriction for his consultation results
* Add new consultation result with diagnostic, consultation, image (where is necessarily) and recipe.
* View consultations results
* Modify consultations results
* Delete consultations results
* Diagnostics management with restriction for his specialty diagnostics:
* Add new diagnostic with name and code
* View diagnostics
* Modify existing diagnostic
* Delete diagnostic

Apart from the functionalities offered by the group’s permission, every user has the possibility to change his password provided by the administrator.

Another system requirement is to send text messages and emails for the following scenarios:

* New appointment is set for a doctor – Application will notify the doctor with a text message
* New consultation result is set for patient – The application will send the consultation results for the patient via e-mail (If the patient is not registered with an e-mail address and only with a phone number will receive a notification text message when his results are ready)

## 3.2 Use case diagram



## 3.3 System architecture

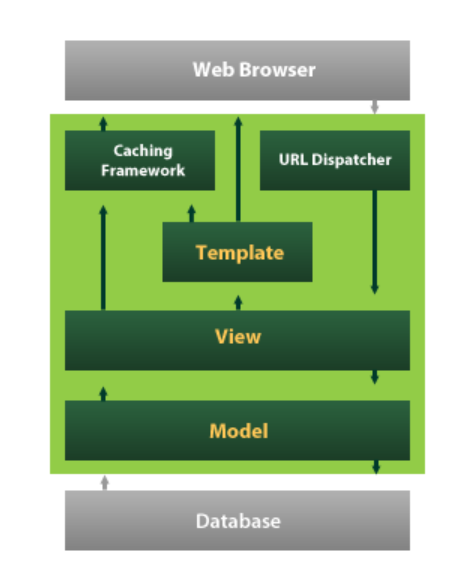
The URL dispatcher (urls.py) maps the requested URL to view function and calls it. If caching is enabled, the view function can check to see if a cached version of the page exists and bypass all further steps, returning the cached version, instead. Note that this page level caching is only available caching option in Django. You can cache more granularly, as well.

The view function (usually in views.py) performs the requested action, which typically involves reading or writing to the database. It may include tasks, as well.

The model (usually in models.py) defines the data in Python and interacts with it. Although typically contained in a relational database (MySQL, PostgreSQL, SQLite), other data storage mechanism are possible as well (XML, text files, LDAP).

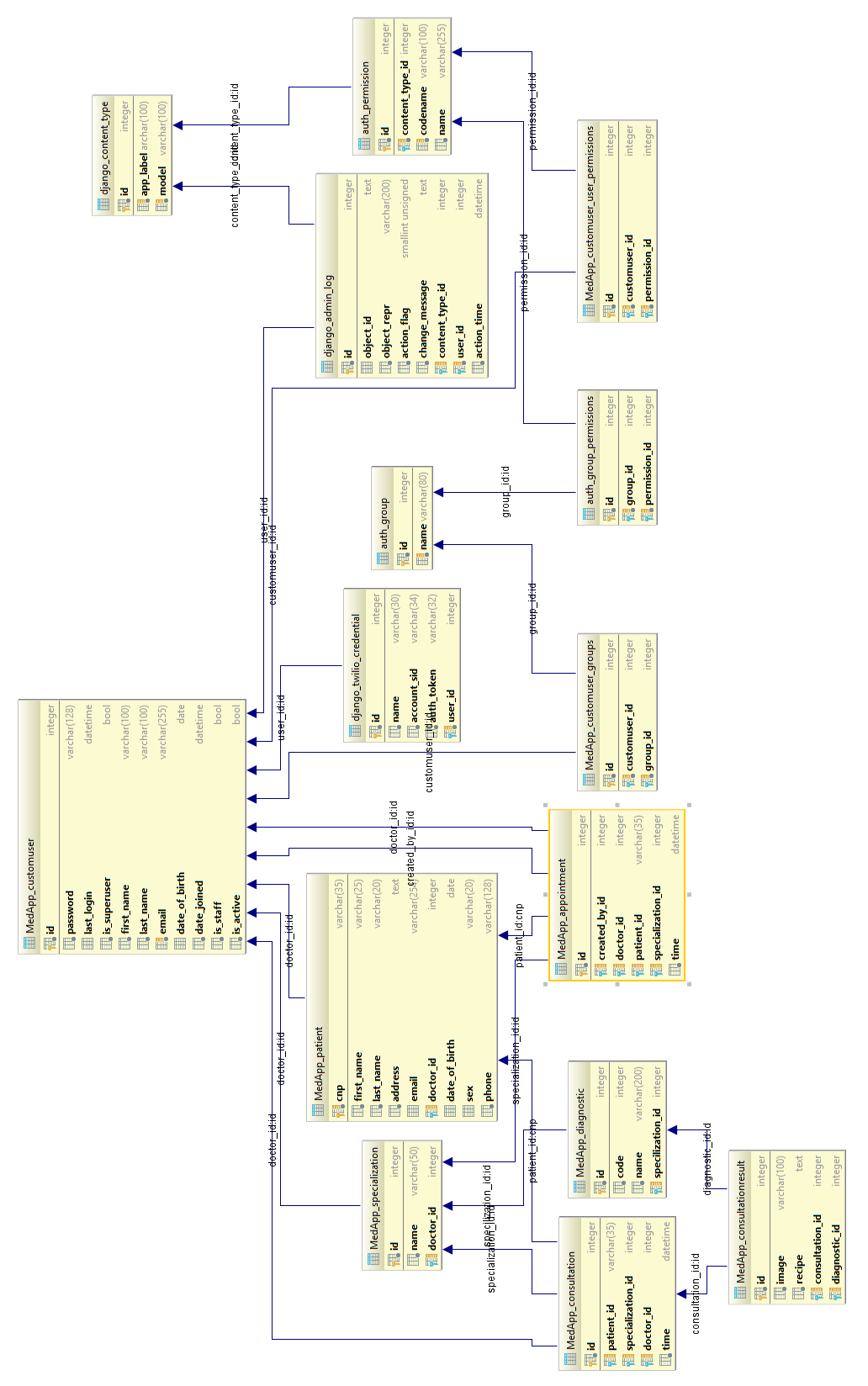
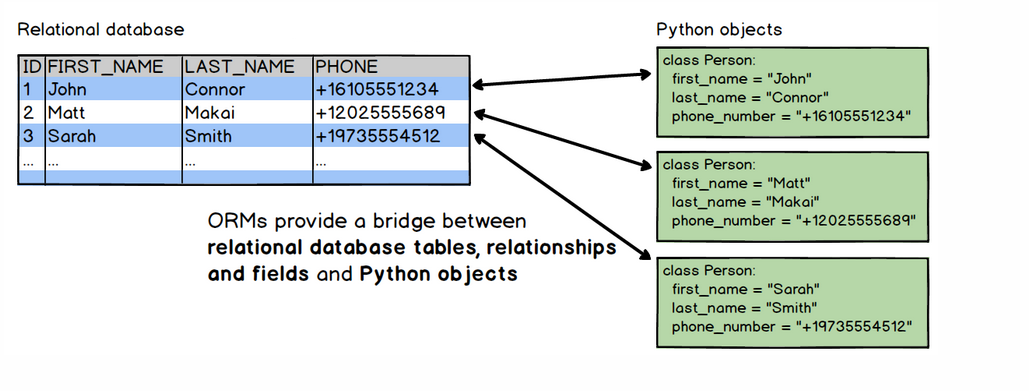
After performing any requested task, the view returns an HTTP response object (usually after passing the data through a template) to the web browser. Optionally the view can save a version of the HTTP response object in the caching system for a specified length of time.

Templates typically return HTML pages. The Django template language offers HTML authors a simple-to-learn syntax while providing all the power needed for presentation logic.



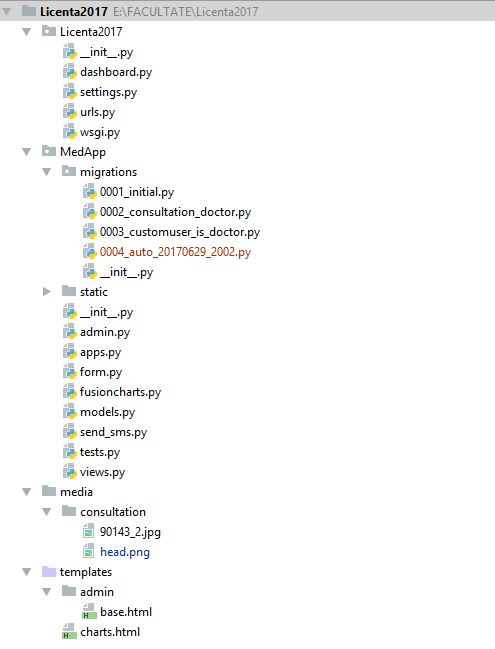
## 3.4 Database design

Working with databases often requires you to get your hands dirty messing about with SQLite. In Django, a lot of this hassle is taken care of for you by Django’s object relational mapping (ORM) functions, and how Django encapsulates databases tables through models. Essentially, a model is a Python object that describes your data model/table. Instead of directly working on the database table via SQLite, all you have to do is manipulate the corresponding Python object.

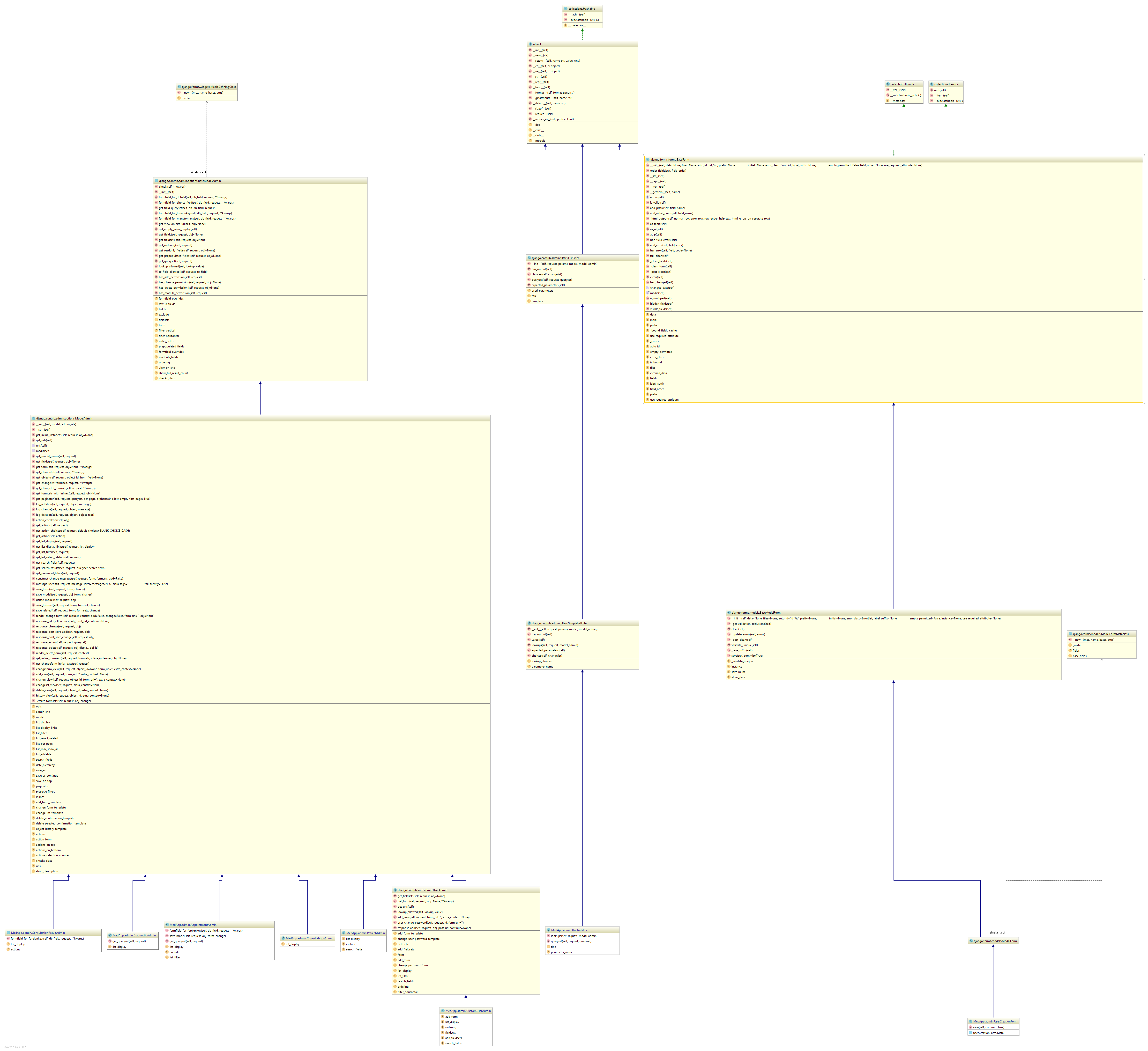


## 3.5 Project structure

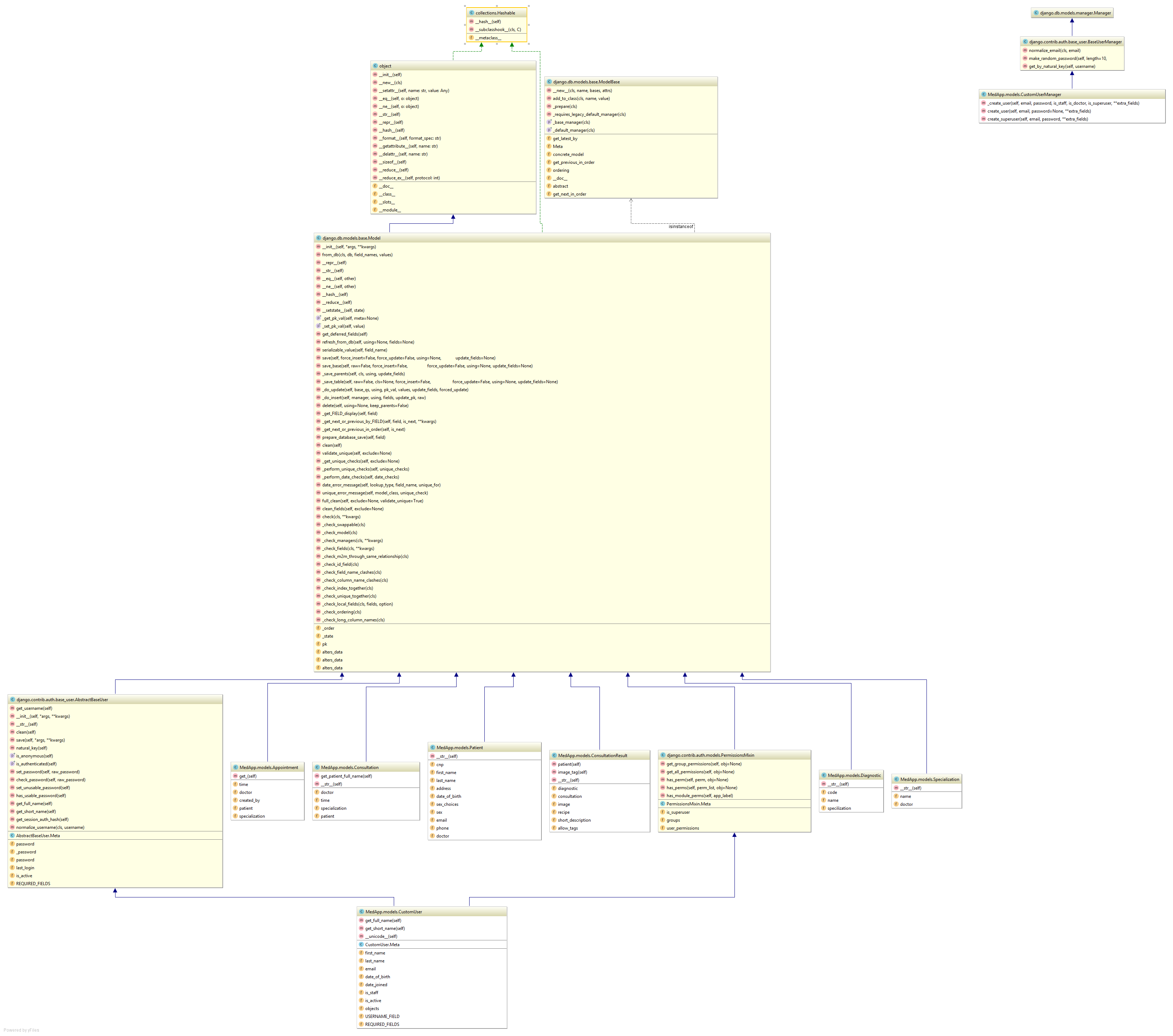
Allows you to pick up, repackage, and reuse individual Django applications for use in other projects. Often it isn’t clear as you are building an app whether or not it is even a candidate for reuse. Building it this way from the start makes it much easier if that time comes. Encourages designing applications for reuse. Project level templates and static files that can, if necessary, override app level defaults.



### 3.5.1 Class Diagram Admin.py



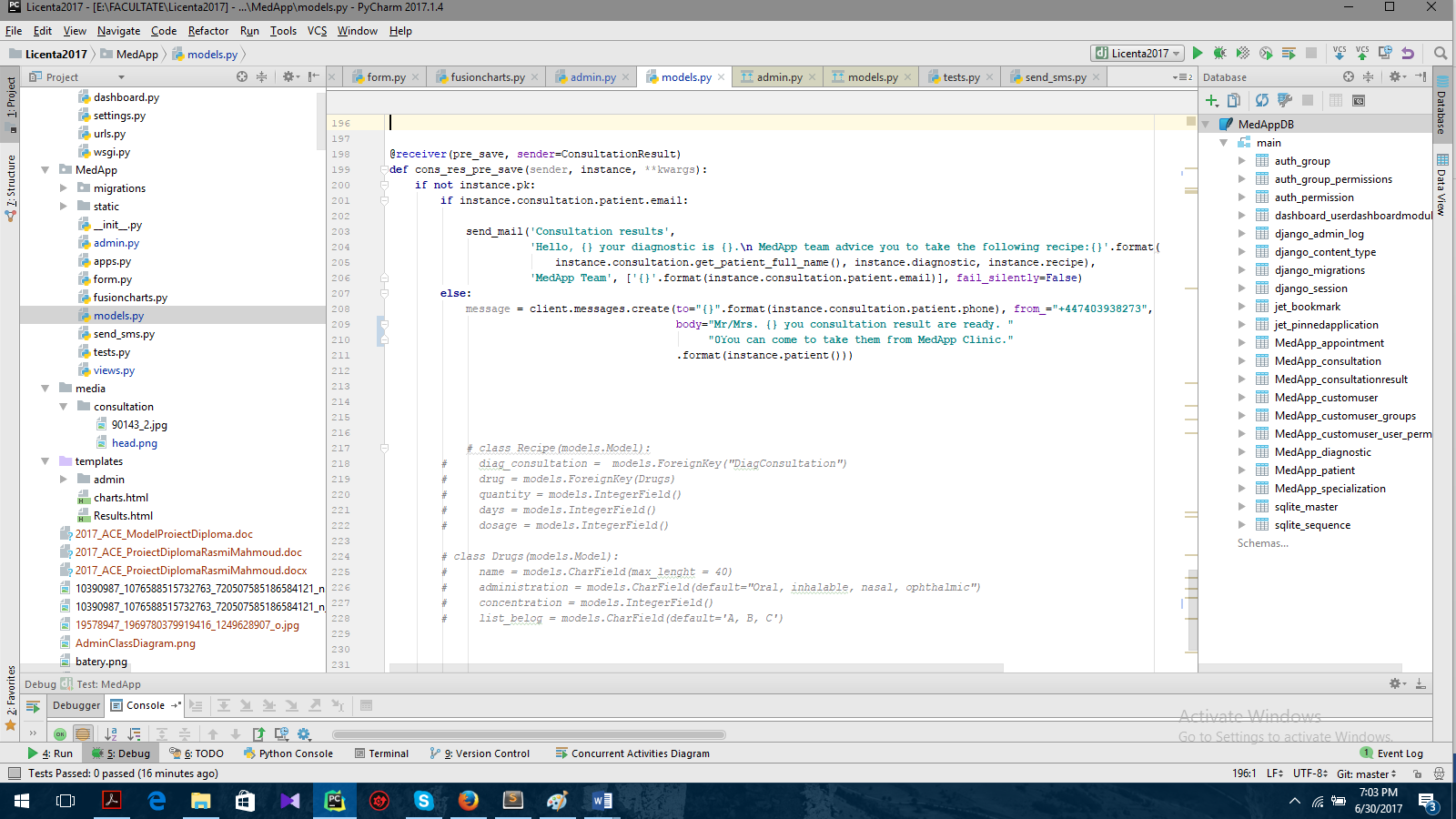
### 3.5.2 Class diagram Models.py



## 3.6 Implementation

The main functionalities will be explained in the following to charters.

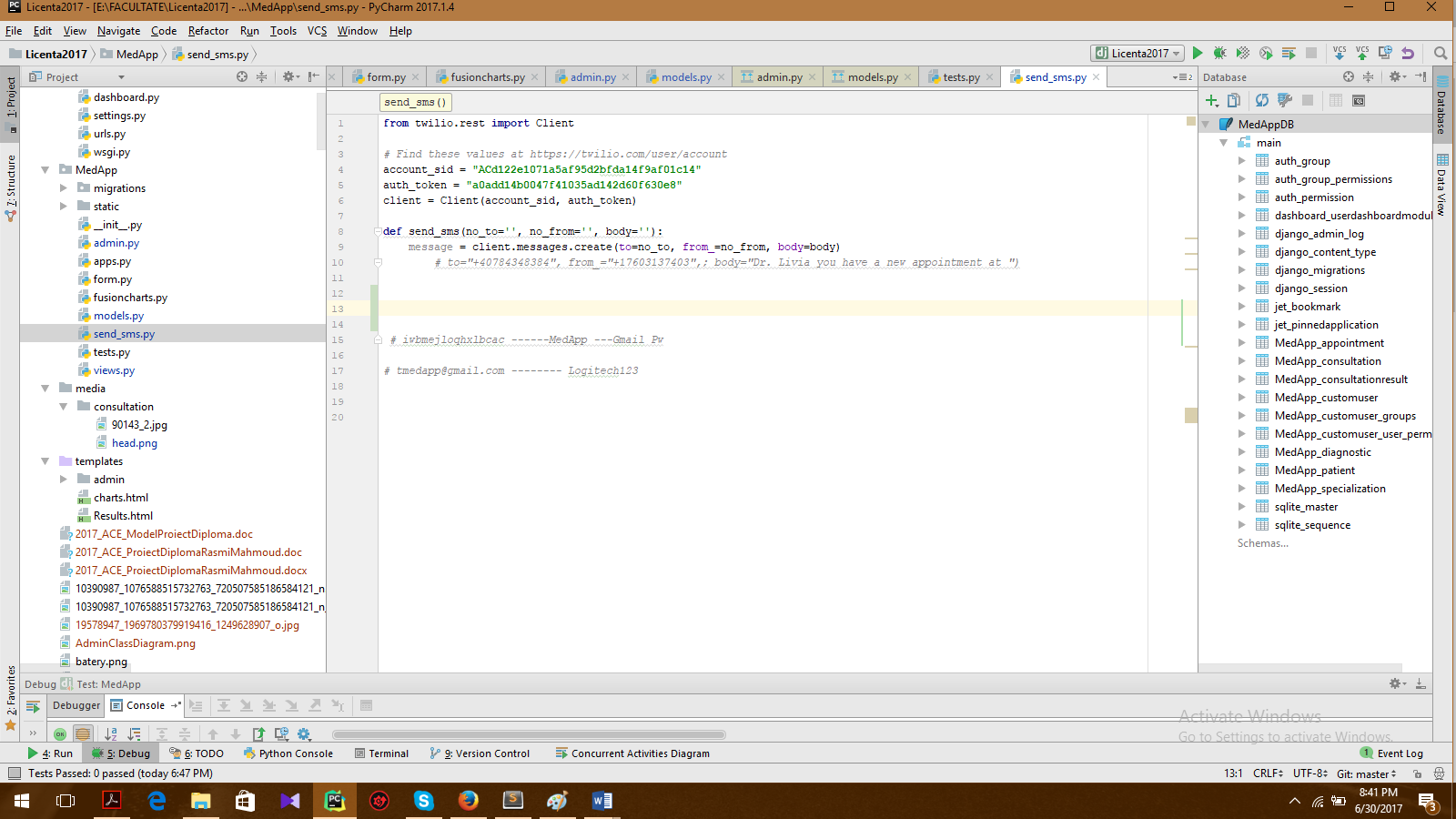
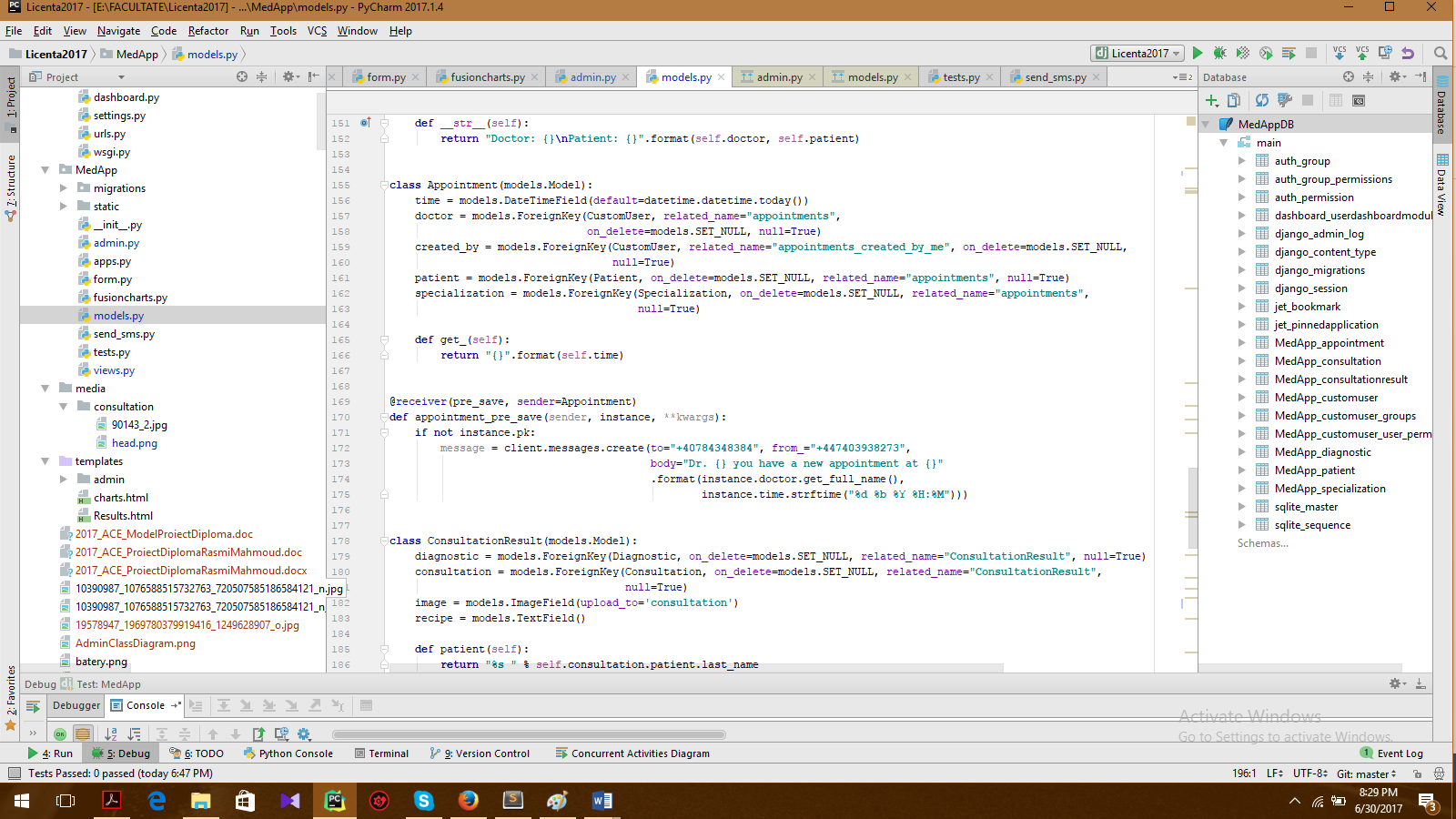
### 3.6.1 Sending e-mail when new results are added



Here the main role goes at Django signals, because many cases when there is a modification in a model’s instance we need execute some action. Django provides us an elegant way to handle with these situations. The signals are utilities that allow us to associate events with actions. We can develop a function that will run when a signal calls it. So, we use the pre\_save() signal and whenever a new entry in the database is saved the signal decouples the application and an e-mail is send with the new diagnostic to the patient.

If the patient is registered with an e-mail address and telephone number the first option for notifying him is to send an e-mail. In cases where the patient is not registered with an e-mail a text message (the function that sends text messages will be explain in the next charters) will be send to his phone number.

### 3.6.2 Sending text message when new appointment is added



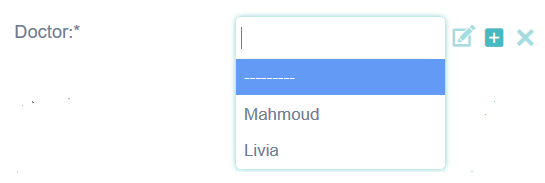
The logic is almost the same as for e-mail send we use Django signals for creating a notify event, but here we will use an API for sending text messages. The API is called Twilio and I presented in the charter dedicated for technologies and frameworks. Sending a message is as simple as POST-ing to the Messages resource, but it requires some parameters:

* The “To” parameter is required in your POST to send the message. “To” parameter is used for the destination phone number, which must respect the international format for phone numbers ( “+” followed by the country code)
* “From” parameter. A Twilio phone number or alphanumeric sender ID enabled for the type of message you wish to send. Phone numbers or short codes purchased from Twilio work here. You cannot (for example) spoof messages from your own cell phone number.
* “Body” parameter. The text of the message you want to send, limited to 1600 characters.

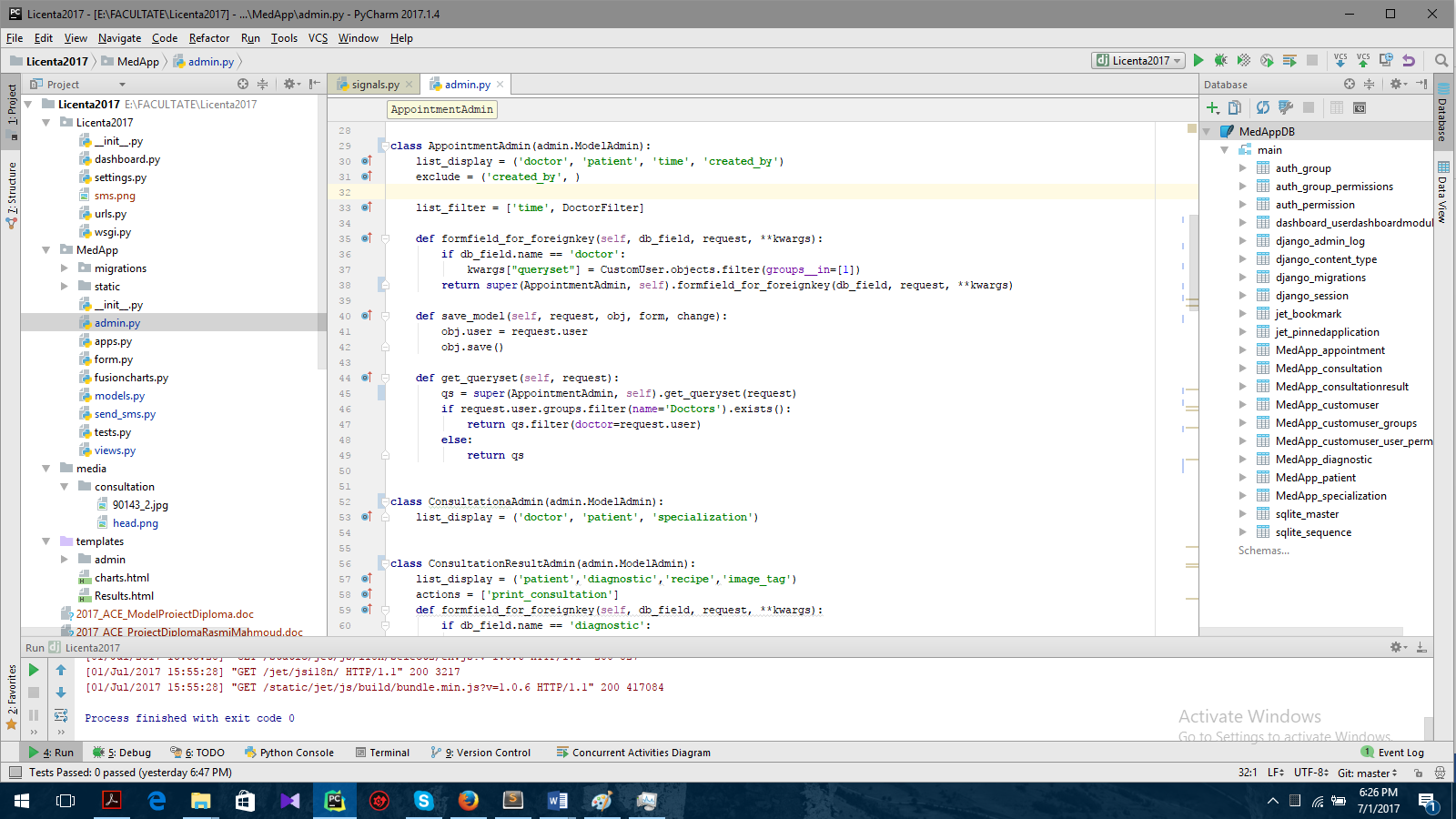


### 3.6.3 Filter for viewing only relevant information based on logged in user

When creating a new appointment only doctors can be selected, not all users.



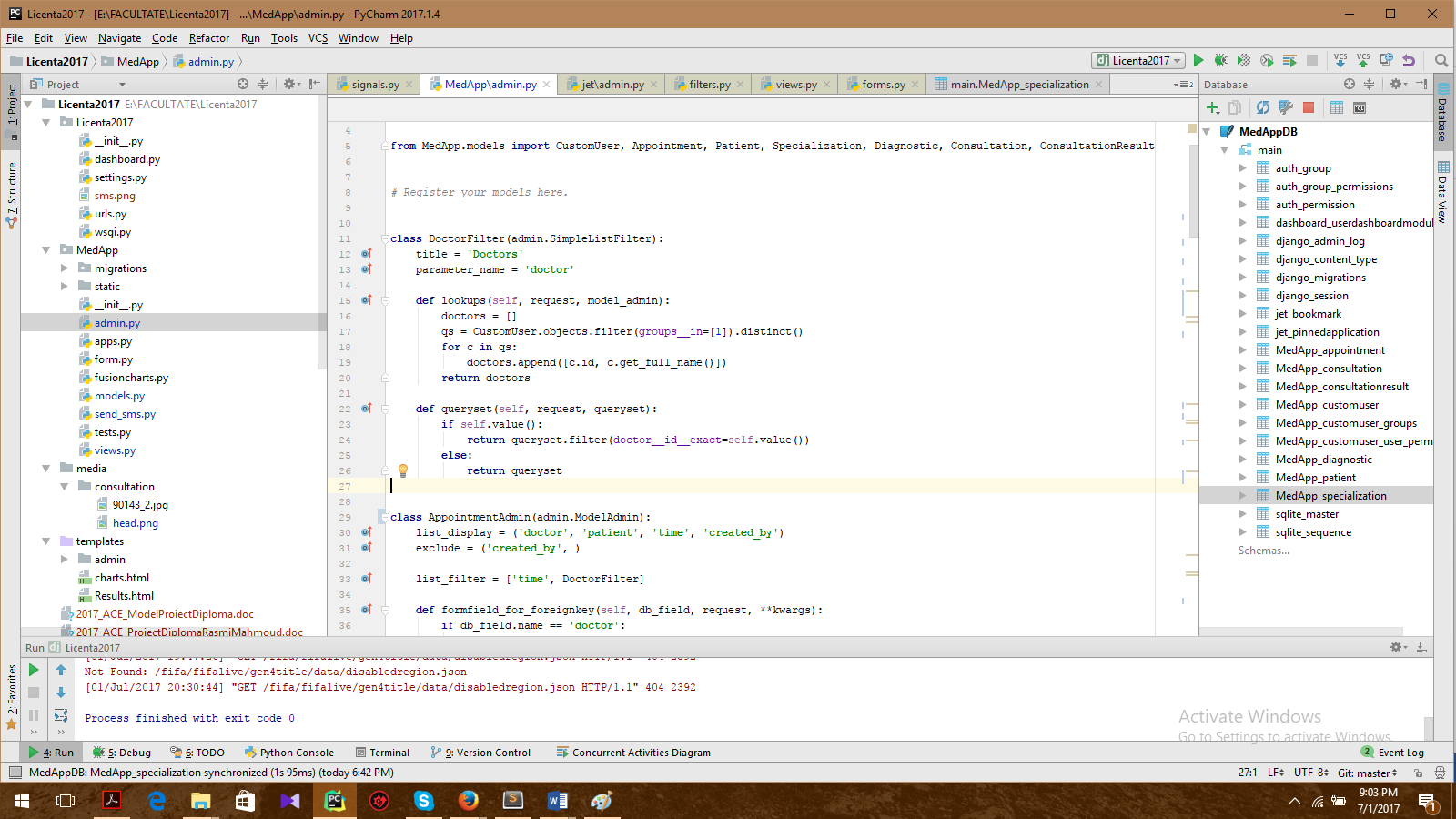
We create a function and then we filter the users to be in our “Doctor” group.



For displaying only the information relevant for the logged in user, we create a function were we verify first if the logged in user is a doctor and if this is true the information will be filtered. If the logged in user is not a doctor the application will display all data.

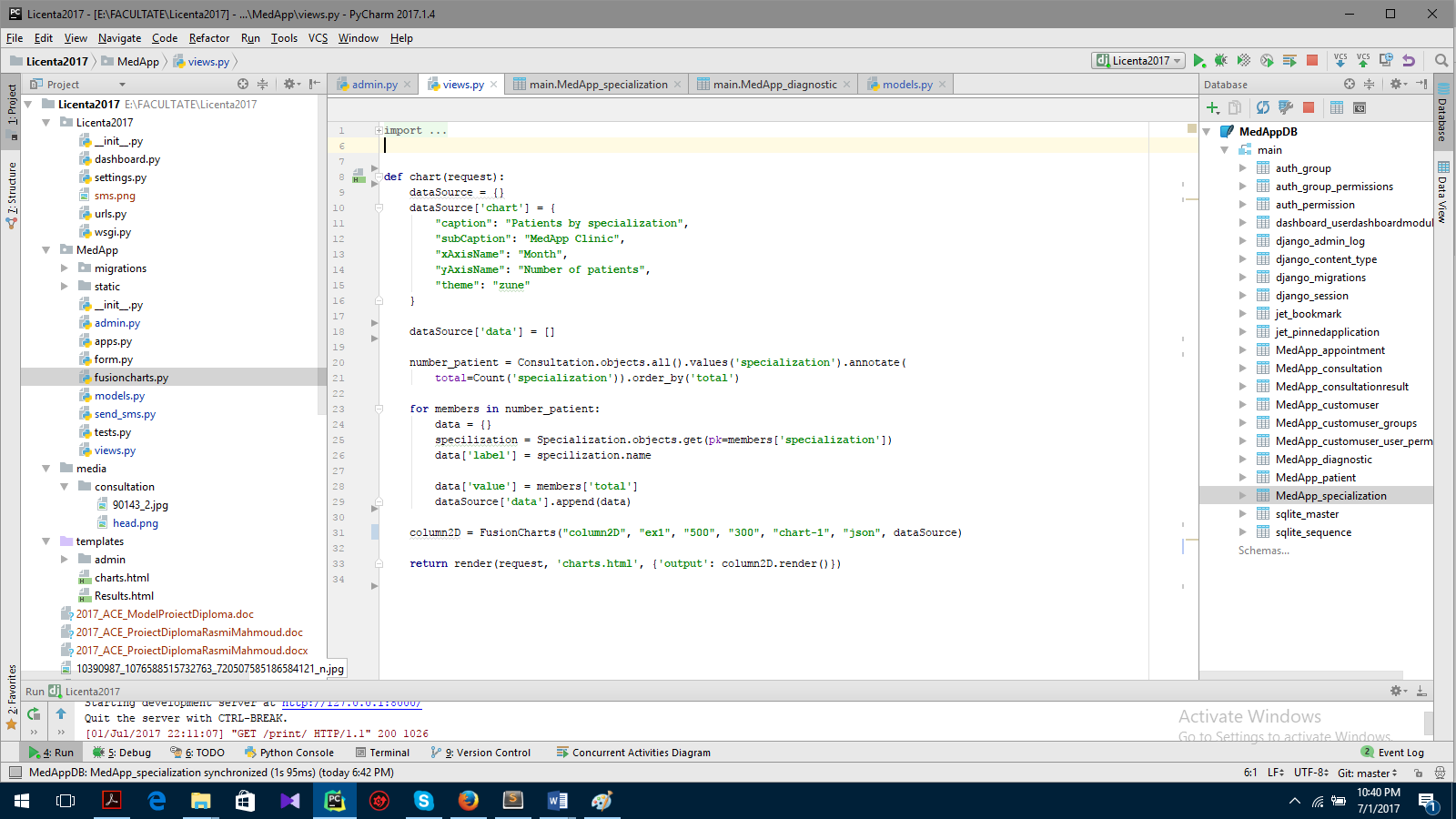


Filter displayed data by doctor. First we search for the users that are in the doctor’s group we store them in a list and then we our query set.

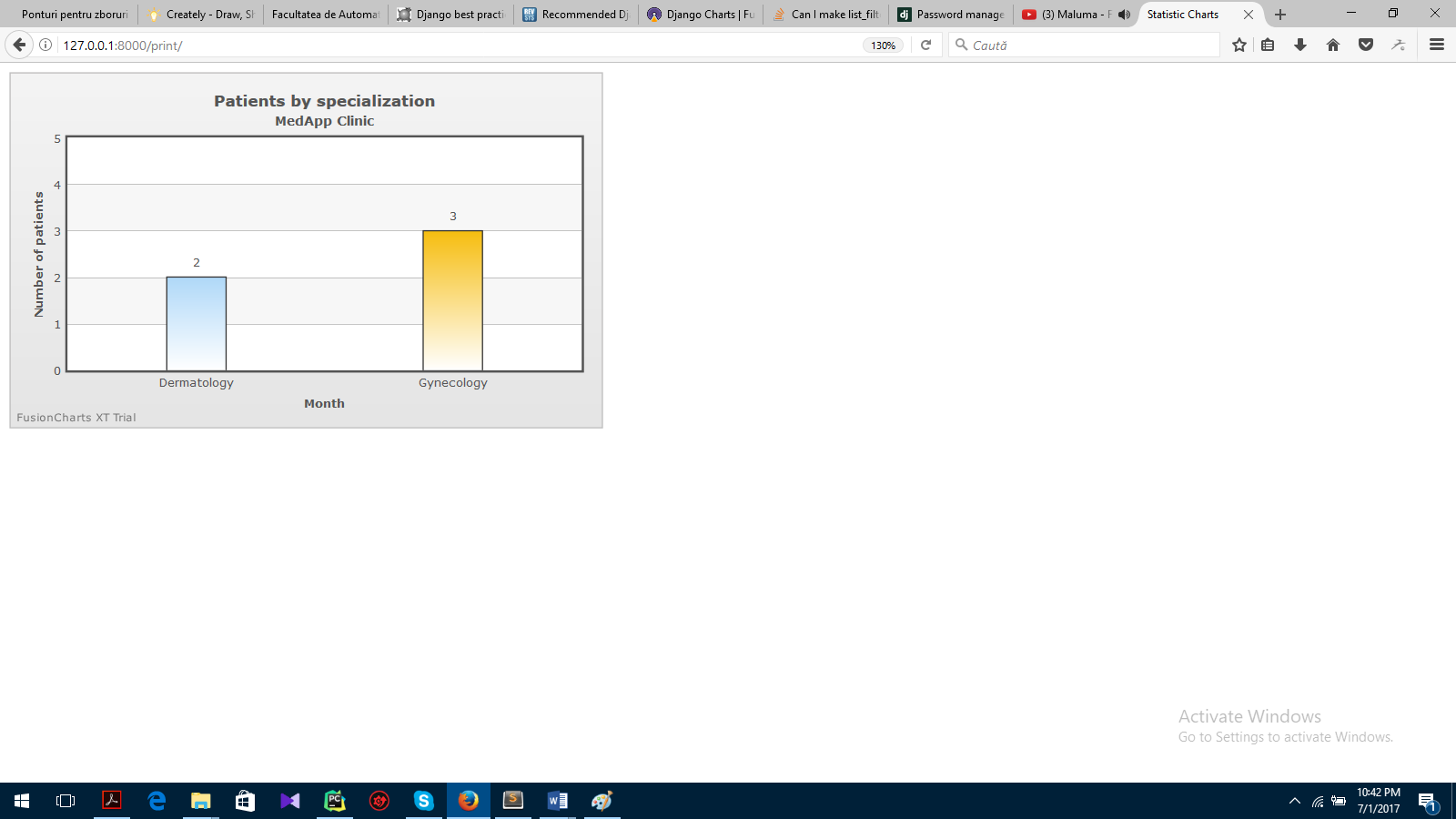


### 3.6.4 Charts

The admin has the possibility to view some statistics based on the clinic activity.



Fusion Charts Free is an open-source FREE flash charting component that can be used to render data-driven animated charts. The “chart” function is defined to load chart from a SQLite database. At first, data is retrieved from the SQLite database. This data is used to create chart data. Chart data is passed to the “dataSource” parameter, as dict, in the form of key-value pairs. The data for the chart should be in an array where each element of the array is a JSON object as “label” and “value” keys. Iterate through the data in `Country` model an d insert in to the “dataSource['data']” list. Creating an object for the Column 2D chart using the FusionCharts class constructor and then returning complete JavaScript and HTML code, which is used to generate chart in the browsers.



### 3.6.5 Django Password management

Django provides a flexible password storage system and uses PBKDF2 by default. The password attribute of a User object is a string in this format:

**<Algorithm>$<iterations>$<salt>$<hash>**

Those are the components used for storing a User’s password, separated by the dollar-sign character and consist of: the hashing algorithm, the number of algorithm iterations (work factor), the random salt, and the resulting password hash. The algorithm is one number of one-way hashing or password storage algorithms Django can use. Iterations describe the number of times the algorithm is run over the hash. Salt is the random seed used and the hash is the result of the one-way function.

One of the features of the Auth module I like the most is password hash upgrading. When a user successfully authenticates, if their password hash is using any other hashing algorithm or iteration count than the first hashing algorithm configured in Django then Django will take the raw password and re-hash it using the first configured hasher.

If you were to create a password in Django 1.10.3 that was hashed with anything other than PBKDF2 and with the default iteration count then the password hash will be upgraded to PBKDF2 and 30,000 iterations.

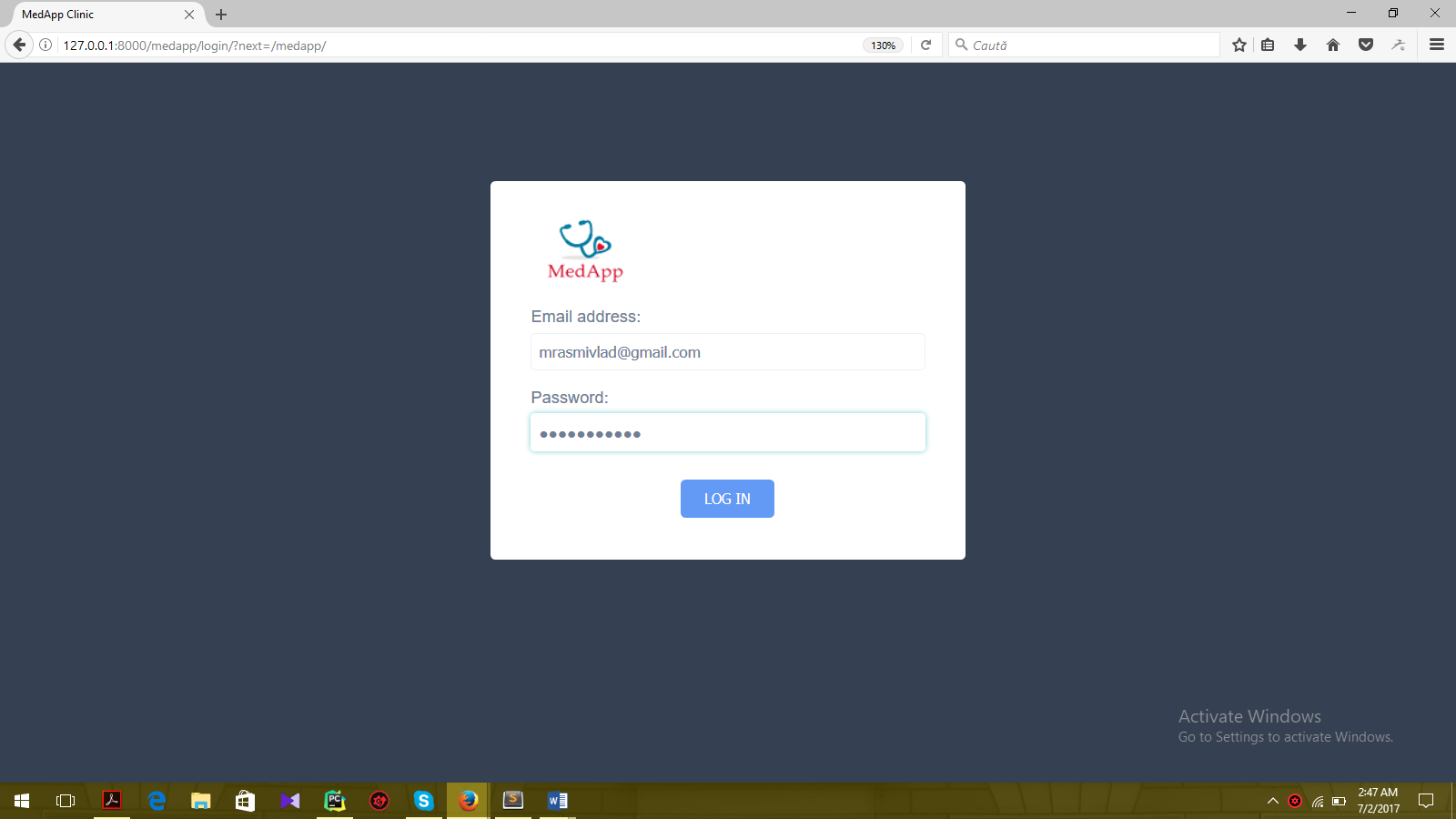


For storing the password in a crypted way we will use a form in admin.py, where we specify that the password must be stored as a password. When using the set\_password method the first (and presumably the strongest) password hasher that Django is configured with will be used. In the case of Django 1.10.3 it's PBKDF2 + HMAC + SHA256 with 30,000 hashing iterations performed on the original password string.



# User manual

The entry point of the application is the login page.



If the credentials are correct the main application page will be loaded. Because we have users with different permissions every login will load a different start page based on the logged in user.

