INFORMATION SYSTEMS IN HEALTH CARE

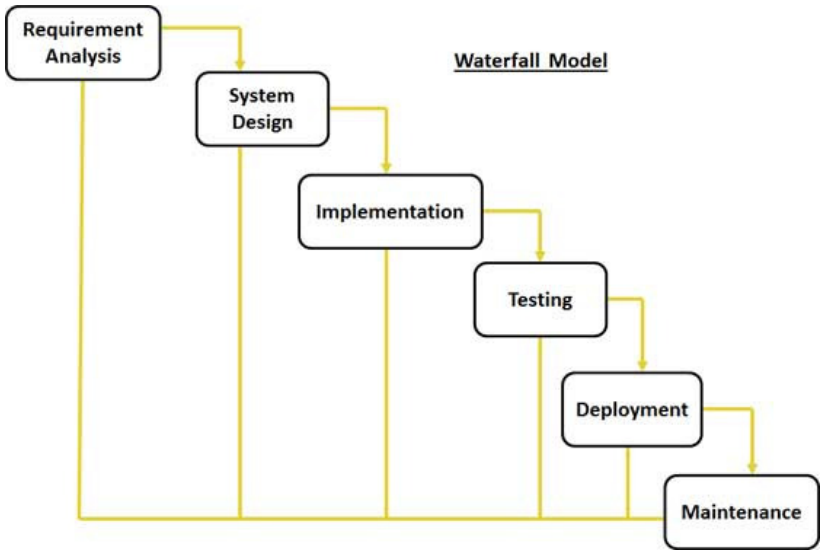
Michel Kana, Ph.D.

Lesson 8 – Winter Term 2014

Schedule

1. **Analysis of software systems**
   a) Software development process
   b) Requirements analysis with UML
   c) Object-oriented modeling with UML
2. **Design of software systems**
   a) Software architecture
   b) User interface design
   c) Design of computing logic

Waterfall software process



- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Requirements analysis

- Consider the following requirements for a hospital information system.
  - *An administrator should have access to the system. He should be able to create new hospital departments and new doctors. Each doctor is assigned to a department. Doctors can log into the system and should be able to create new patients, given their name, age and birth number. Doctors should be able to create a new visit, when a patient has to be hospitalized. The visit start date is recorded. A patient is hospitalized in the doctor's department. Doctors can end the visit by entering a visit end date corresponding to the time when the patient leaves the hospital. Doctors can view patients' list and visits history.*

Requirements analysis - actors

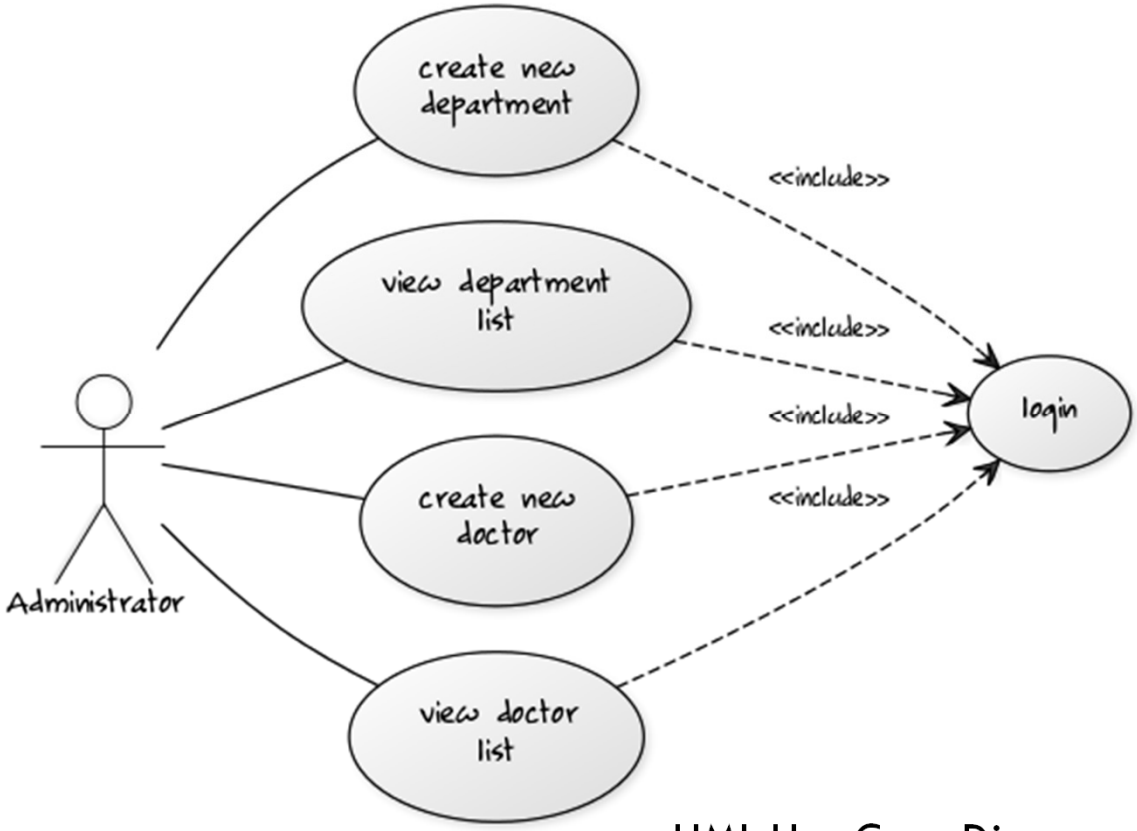
- **Actors** are internal or external agents who interact with the system.
  - *An administrator should have access to the system. He should be able to create new hospital departments and new doctors. Each doctor is assigned to a department. Doctors can log into the system and should be able to create new patients, given their name, age and birth number. Doctors should be able to create a new visit, when a patient has to be hospitalized. The visit start date is recorded. A patient is hospitalized in the doctor's department. Doctors can end the visit by entering a visit end date corresponding to the time when the patient leaves the hospital. Doctors can view patients' list and visits history.*

Requirements analysis – use cases

- A **Use Case** describes a scenario of interaction between the actors and the system.
  - *An administrator should have access to the system. He should be able to create new hospital departments and new doctors. Each doctor is assigned to a department. Doctors can log into the system and should be able to create new patients, given their name, age and birth number. Doctors should be able to create a new visit, when a patient has to be hospitalized. The visit start date is recorded. A patient is hospitalized in the doctor's department. Doctors can end the visit by entering a visit end date corresponding to the time when the patient leaves the hospital. Doctors can view patients' list and visits history.*

Requirements analysis – use cases

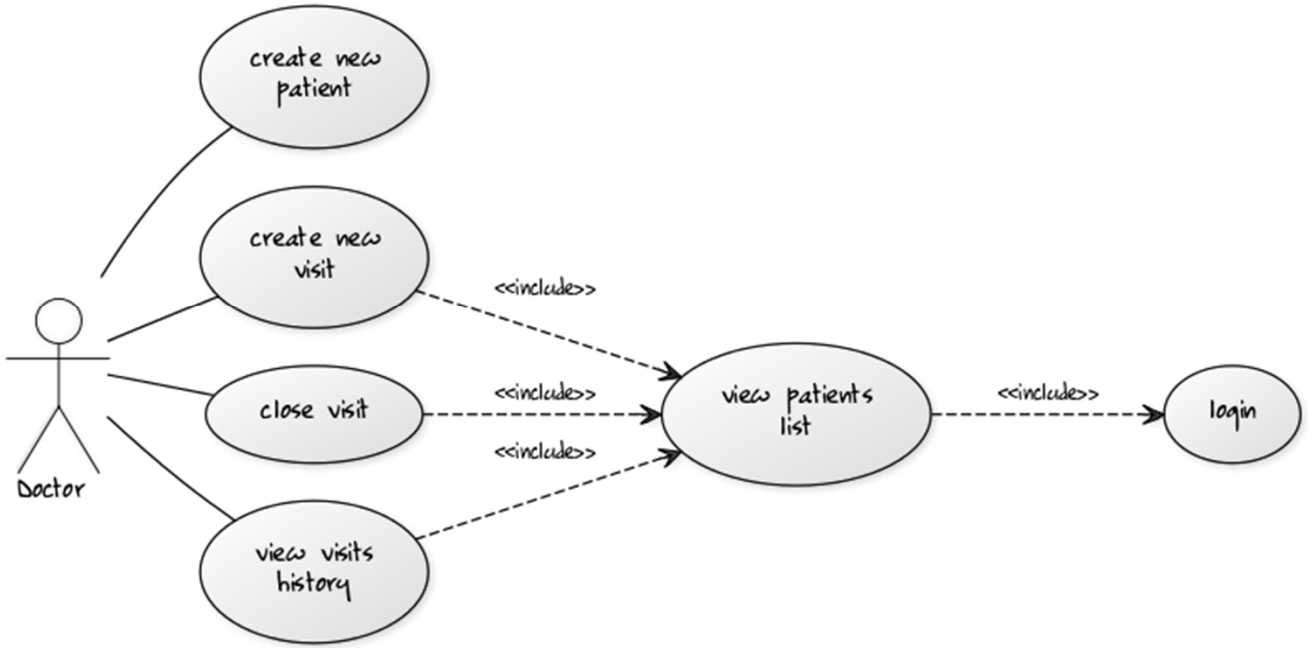
- Actor **Administrator**
  - ***login***
  - ***create a new department***
  - ***view departments list***
  - ***create a new doctor***
  - ***view doctors list***



UML Use Case Diagram

Requirements analysis – use cases

- Actor ***Doctor***
  - ***Log in***
  - ***Register a new patient***
  - ***View list of patients***
  - ***Create a new visit in hospital***
  - ***End a visit***
  - ***View list of hospitalizations***

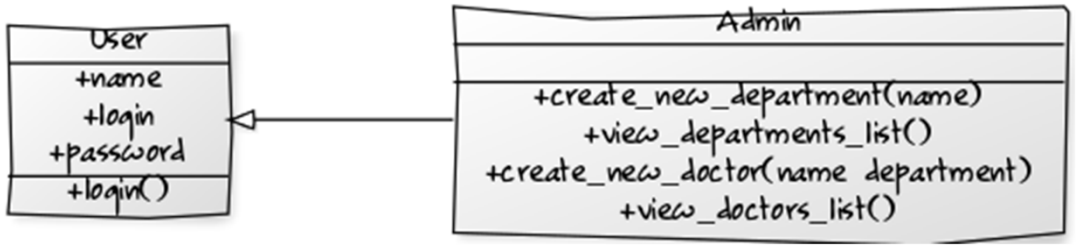


System design – objects and classes

- An *Object* is an independent entity characterized by its attributes and operations. *A Class* is a collection of objects that share a common structure and behaviour.
  - An administrator should have access to the system. He should be able to create new hospital departments and new doctors. Each doctor is assigned to a department. Doctors can log into the system and should be able to create new patients, given their name, age and birth number. Doctors should be able to create a new visit, when a patient has to be hospitalized. The visit start date is recorded. A patient is hospitalized in the doctor's department. Doctors can end the visit by entering a visit end date corresponding to the time when the patient leaves the hospital. Doctors can view patients' list and visits history.

System design – classes

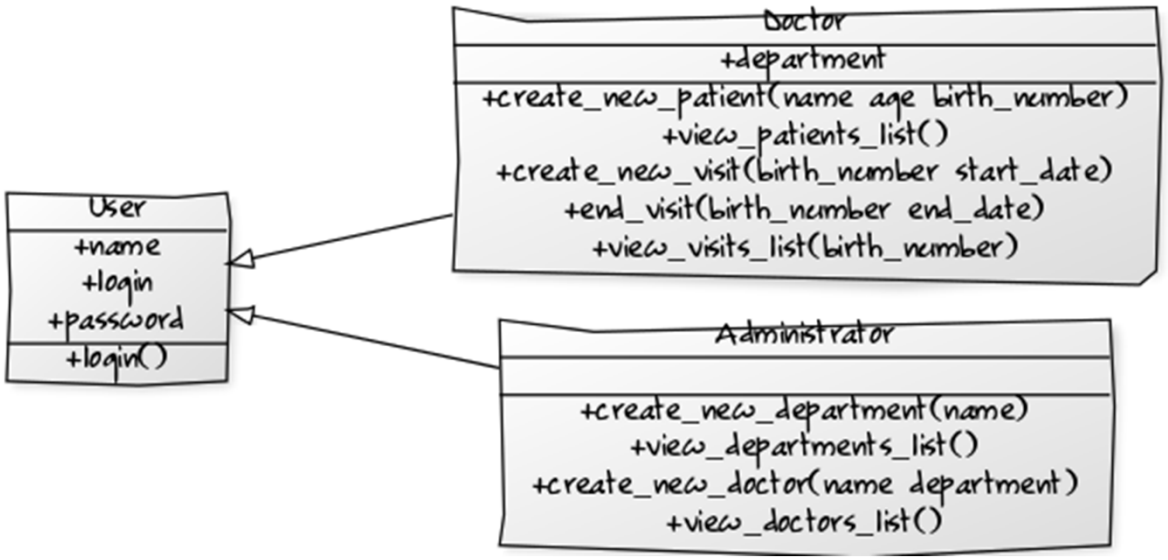
- **User**
  - Atributes:
    - name
    - login
    - password
  - Operations
    - login
- **Administrator**
  - Operations
    - Create a new hospital department
    - View list of departments
    - Create a new doctor
    - View list of doctors



UML Class Diagram

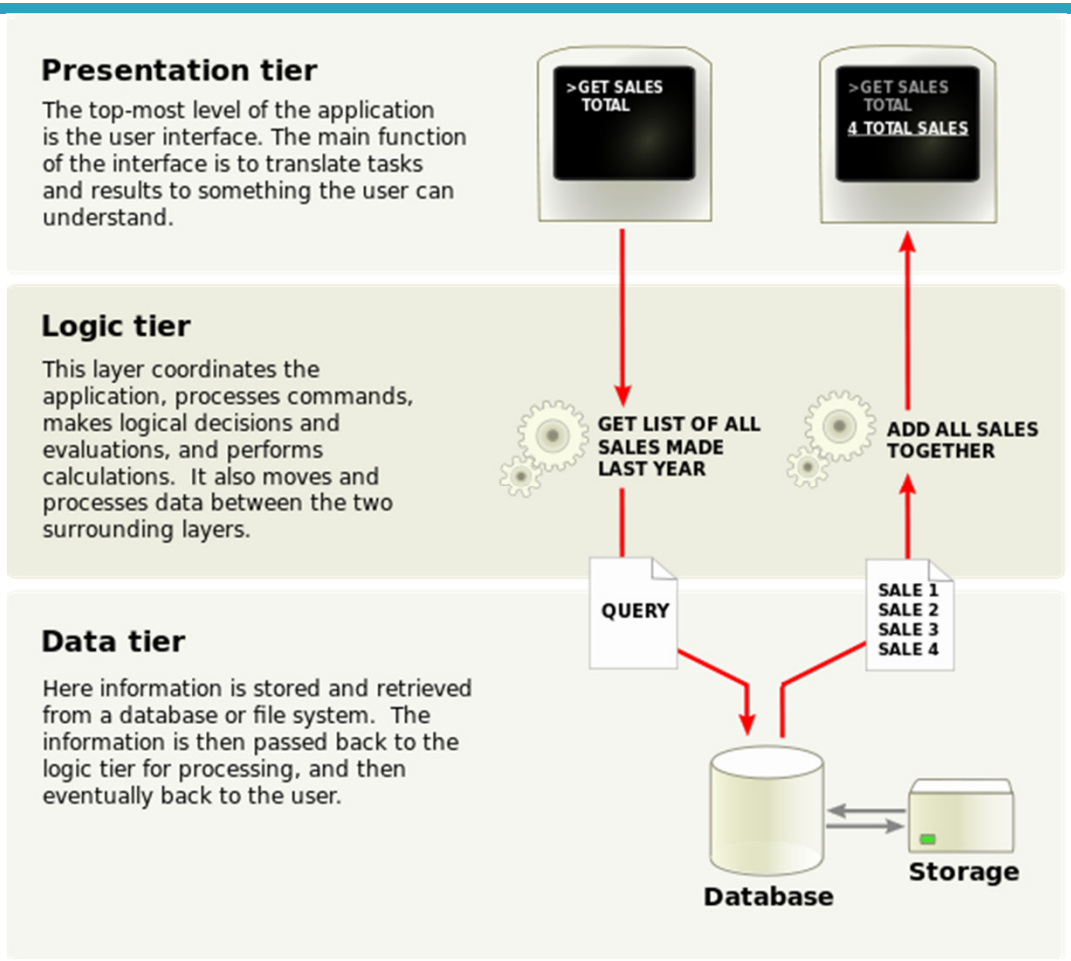
System design – classes

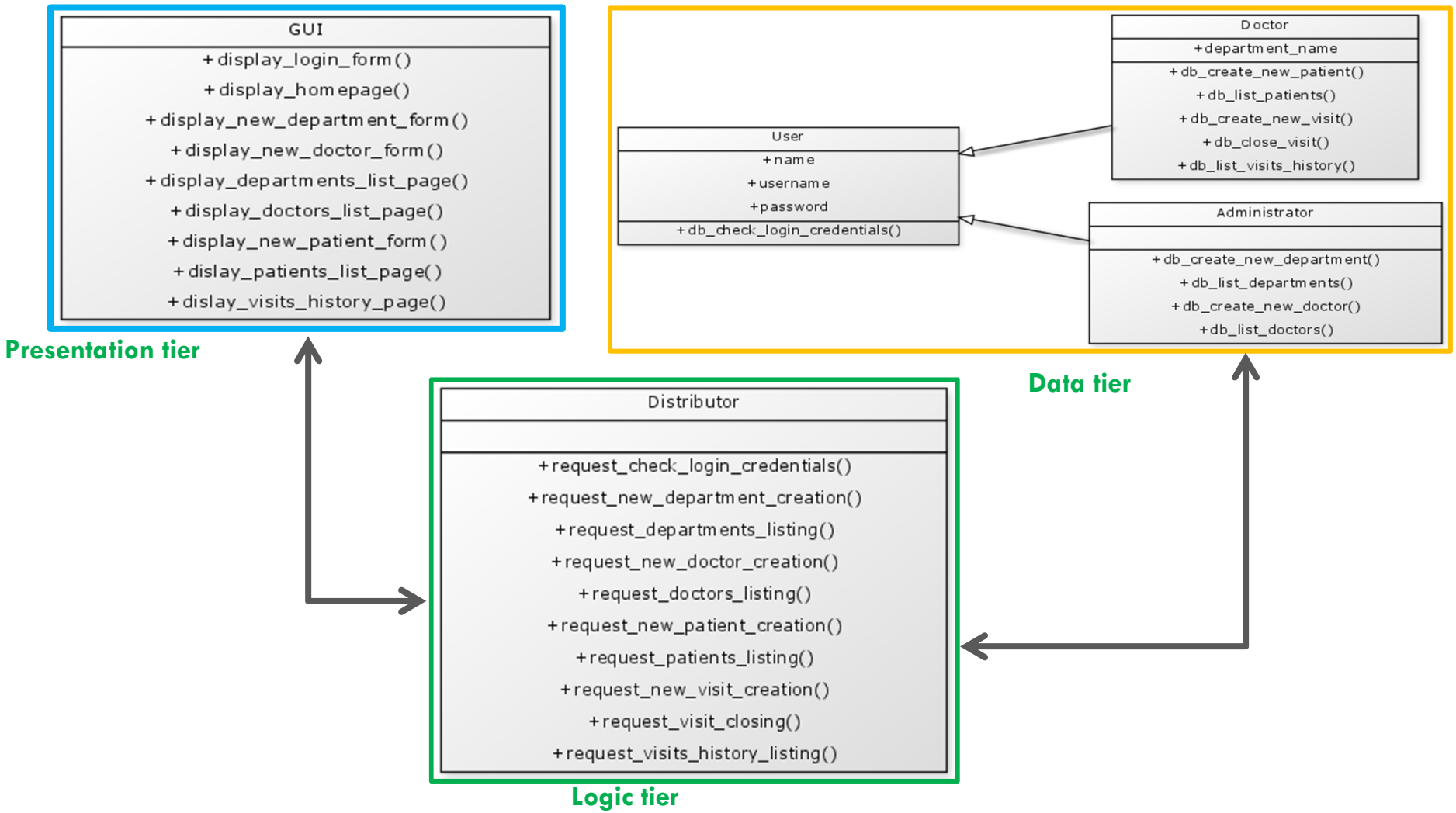
- **User**
  - Atributes:
    - name
    - login
    - password
  - Operations
    - login
- **Doctor**
  - Atributes:
    - Department
  - Operations
    - Create a new hospital department
    - View list of departments
    - Create a new doctor
    - View list of doctors



UML Class Diagram

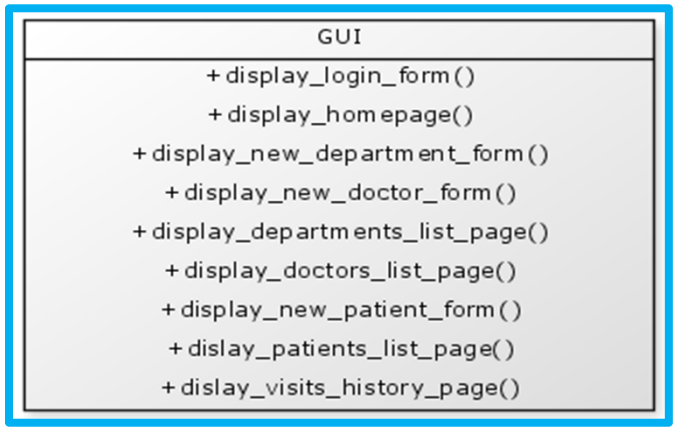
System design – architecture




GUI
+display_login_form()
+display_homepage()
+display_new_department_form()
+display_new_doctor_form()
+display_departments_list_page()
+display_doctors_list_page()
+display_new_patient_form()
+dislay_patients_list_page()
+dislay_visits_history_page()
Presentation tier
Doctor
+department_name
+db_create_new_patient()
+db_list_patients()
+db_create_new_visit()
+db_close_visit()
+db_list_visits_history()
User
+name
+username
+password
+db_check_login_credentials()
Administrator
+db_create_new_department()
+db_list_departments()
+db_create_new_doctor()
+db_list_doctors()
Data tier
Distributor
+request_check_login_credentials()
+request_new_department_creation()
+request_departments_listing()
+request_new_doctor_creation()
+request_doctors_listing()
+request_new_patient_creation()
+request_patients_listing()
+request_new_visit_creation()
+request_visit_closing()
+request_visits_history_listing()
Logic tier

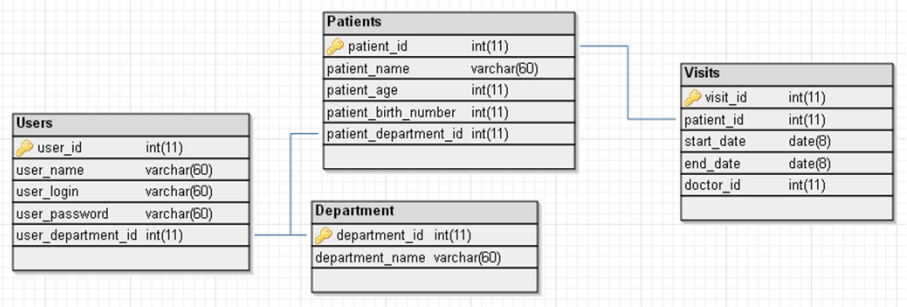
System design – user interface

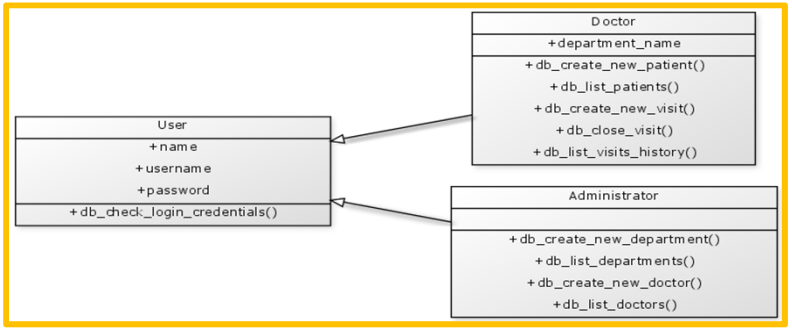
- Administrator
  - Login form
  - Home page
  - Hospital department creation form
  - Doctor creation form
  - Departments listing page
  - Doctors listing page
  - System messages
- Doctor
  - Login form
  - Home page
  - Patient registration form
  - Patients listing page with options for starting and ending hospitalization
  - Hospitalizations listing page
  - System messages



System design – database

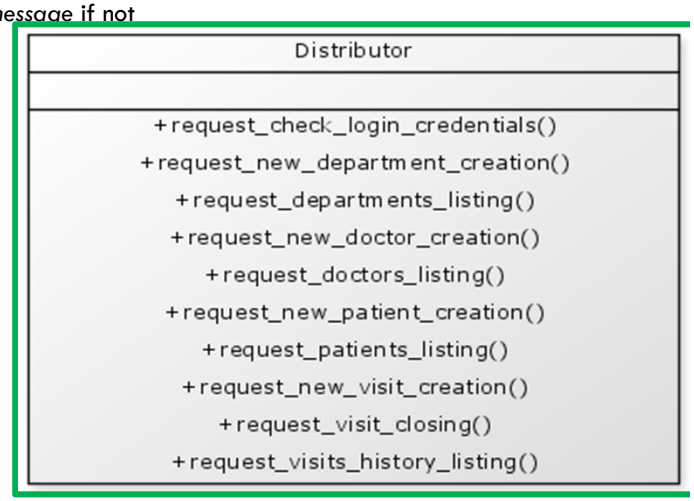
- **User**
  - check login credentials
    - returns the name of the user found in the table *User(name, username, password, department) for a given username and password*
- **Administrator**
  - create new department
    - insert a new record in the table *Department (department_id, department_name)*
  - list departments
    - returns the content of the table *Department (department_id, department_name)*
  - create new doctor
    - insert a new record in the table *Doctor (doctor_id, doctor_name, department_name)*
  - list doctors
    - returns the content of the table *Doctor (doctor_id, doctor_name, department_name)*
- **Doctor**
  - create new patient
    - insert a new record in the table *Patient (patient_name, patient_age, patient_birth_number)*
  - list patients
    - returns the content of the table *Patient (patient_name, patient_age, patient_birth_number)*
  - create new visit
    - insert a new record in the table *Visits (patient_birth_number, start_date, end_date, doctor_id)*
  - End visit
    - Update the end_date in the table *Visits (patient_birth_number, start_date, end_date, doctor_id)*
  - list patient visits
    - returns the content of the table *Visits (patient_birth_number, start_date, end_date, doctor_id)* for a given patient

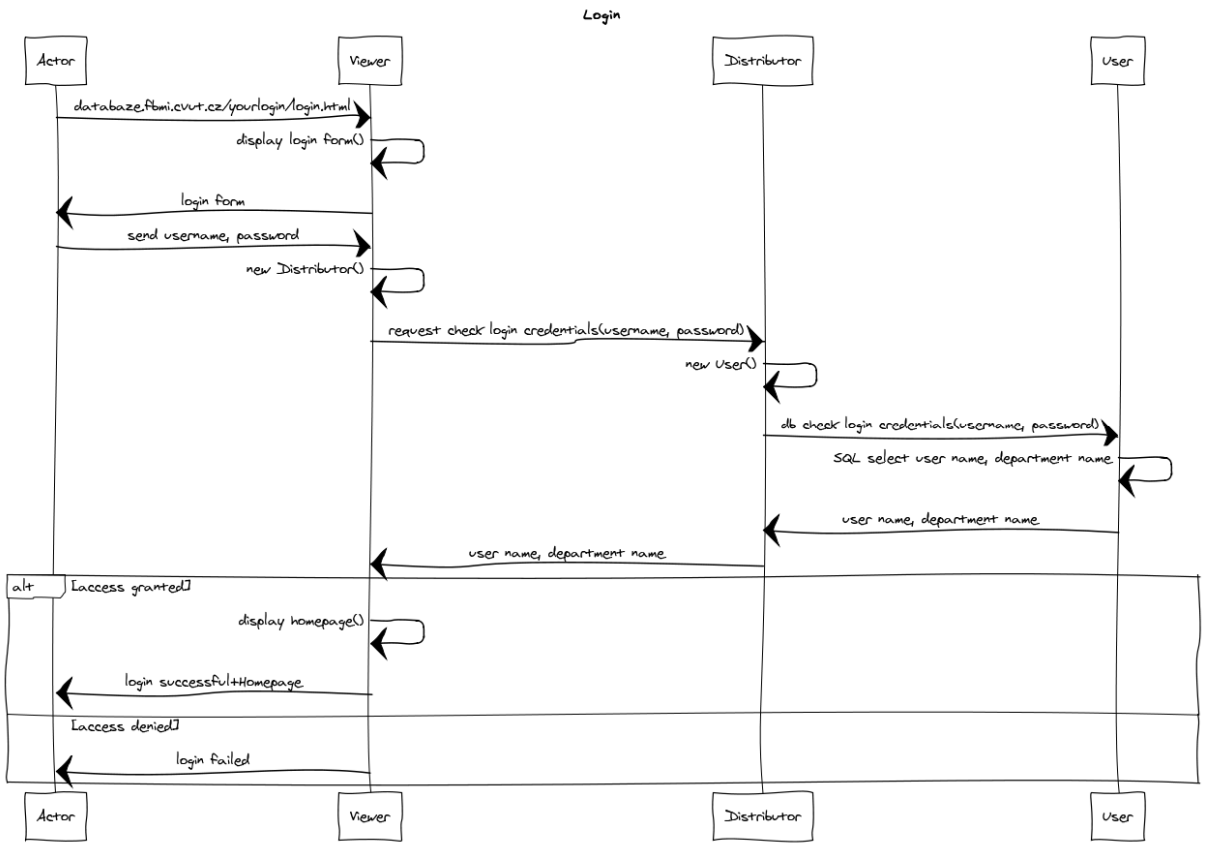


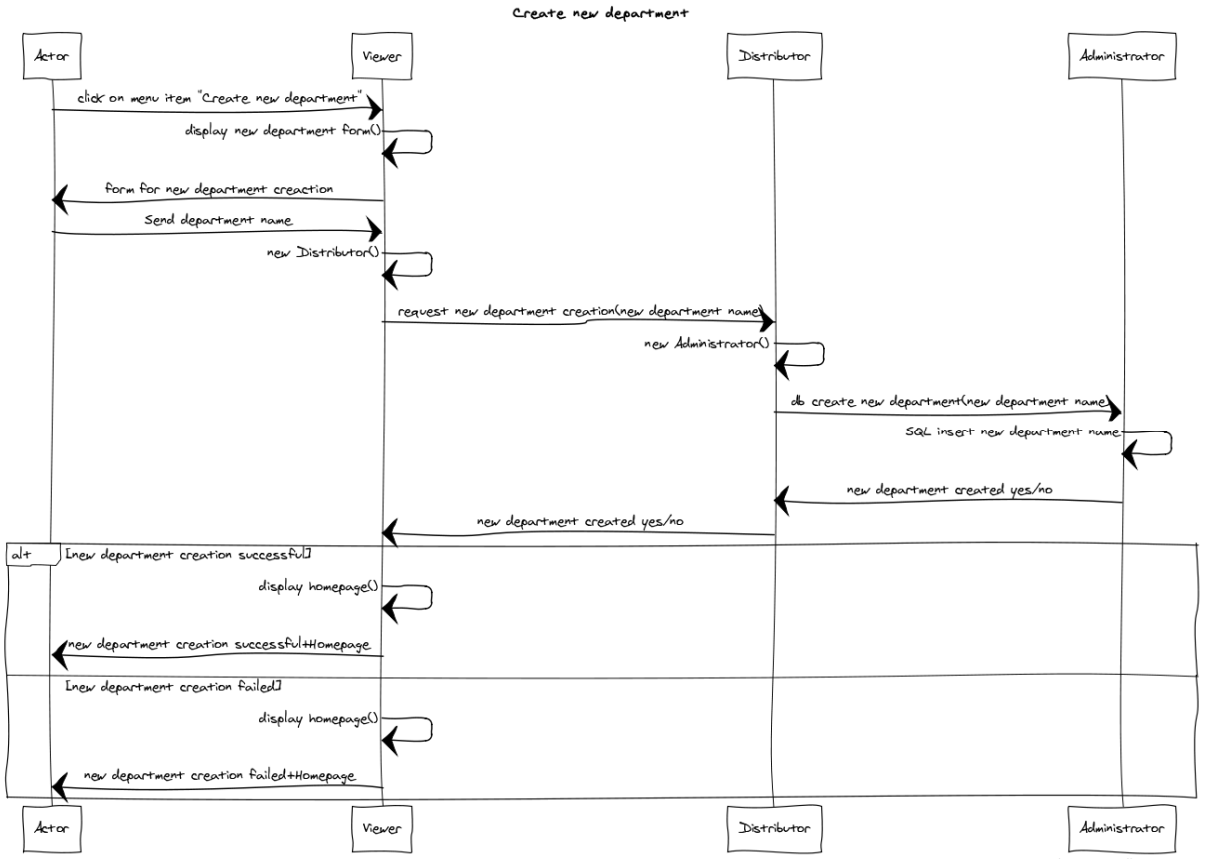


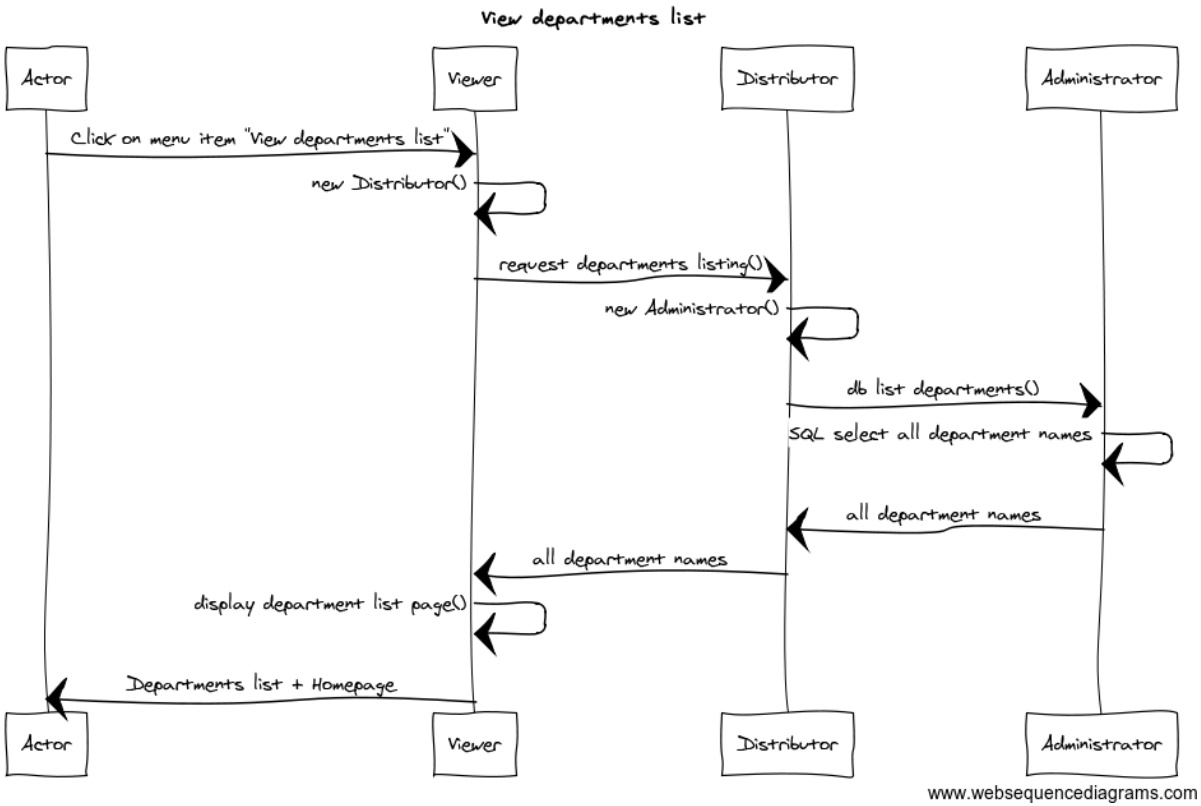
System design – application logic - administrator

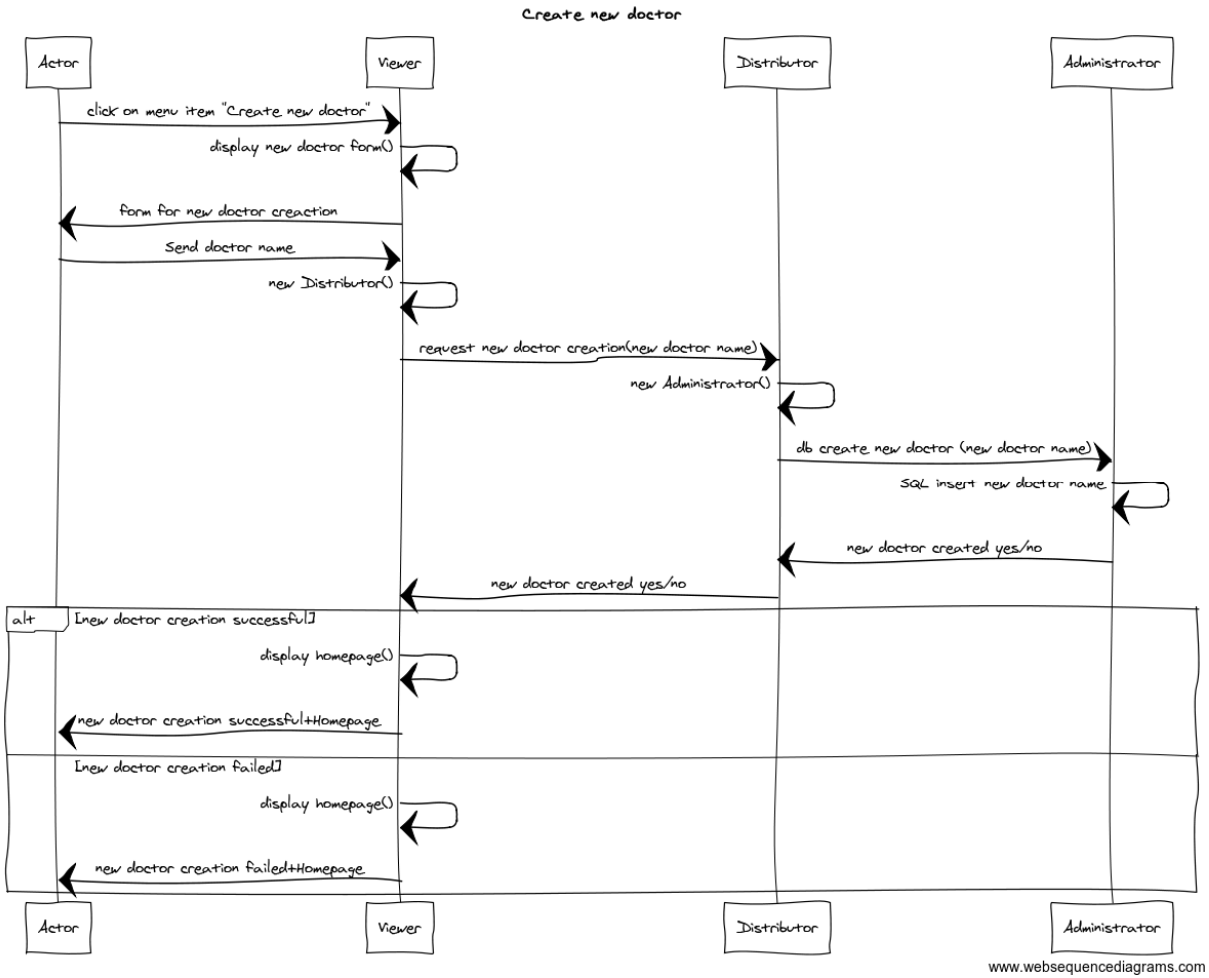
- **login**
  1. create an object from the class *GUI* and display the *login form*
  2. get credentials (username and password) from the *login form*
  3. create an object from the class *User* and verify access rigths
  4. create an object from the class *GUI* and display the *home page* if access is granted or a *system message* if not
- **create a new hospital department**
  1. create an object from the class *GUI* and display the *department creation form*
  2. get the department name from the *department creation form*
  3. create an object from the class *Administrator* and store the name of the new department
  4. create an object from the class *GUI* and display the *departments listing page*
- **view list of hospital departments**
  1. create an object from the class *Administrator* and fetch the list of departments
  2. create an object from the class *GUI* and display the *departments listing page*
- **create a new doctor**
  1. create an object from the class *GUI* and display the *doctor creation form*
  2. get the doctor name and doctor department from the *doctor creation form*
  3. create an object from the class *Administrator* and store the name and department of the new doctor
  4. create an object from the class *GUI* and display the *doctor listing page*
- **view list of doctors**
  1. create an object from the class *Administrator* and fetch the list of doctors
  2. create an object from the class *GUI* and display the *doctors listing page*

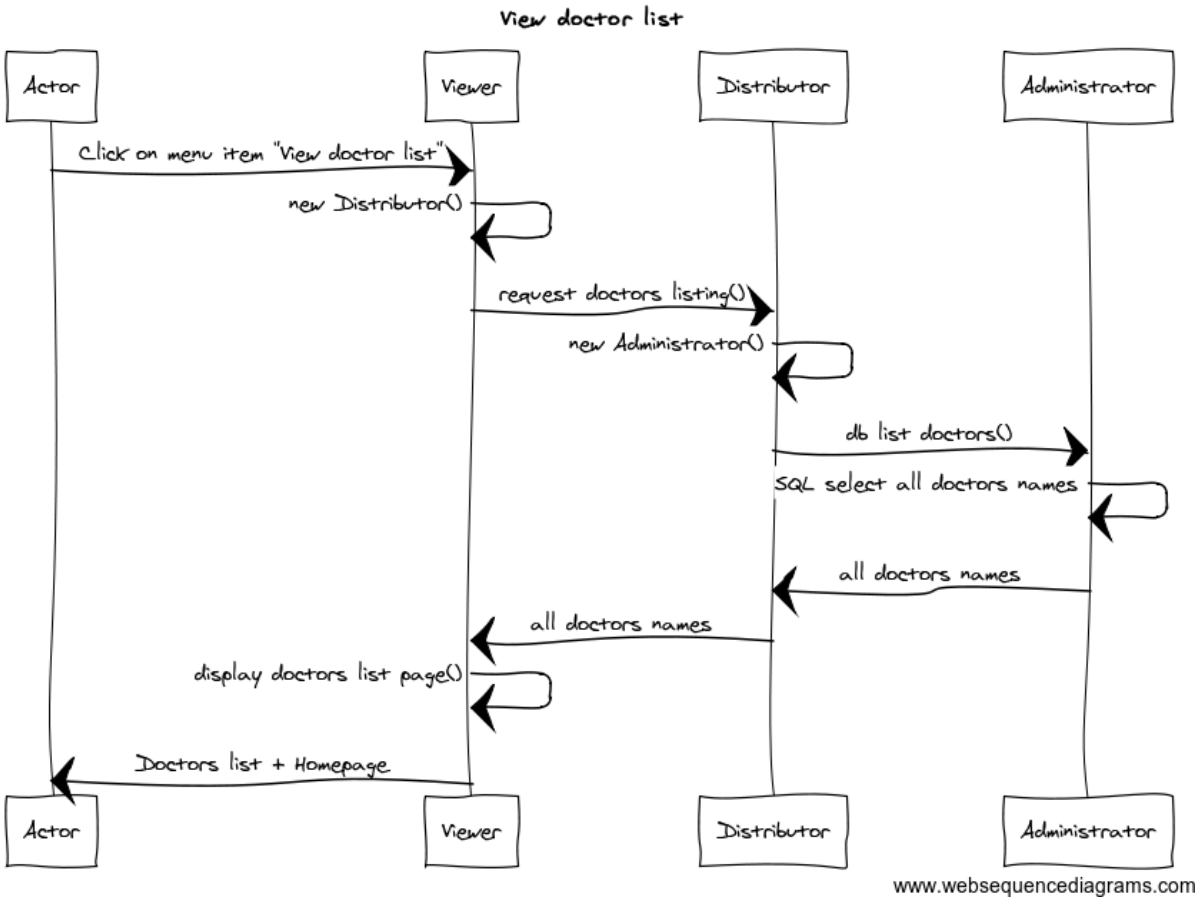



Login
Actor
Viewer
Distributor
User
databaze.fbmi.cvut.cz/yourlogin/login.html
display login form()
login form
send username, password
new Distributor()
request check login credentials(username, password)
new User()
db check login credentials(username, password)
SQL select user name, department name
user name, department name
user name, department name
alt
[access granted]
display homepage()
login successful+Homepage
[access denied]
login failed
Actor
Viewer
Distributor
User
www.websequencediagrams.com


Create new department
Actor
Viewer
Distributor
Administrator
click on menu item "Create new department"
display new department form()
form for new department creation
Send department name
new Distributor()
request new department creation(new department name)
new Administrator()
db create new department(new department name)
SQL insert new department name
new department created yes/no
new department created yes/no
alt
[new department creation successful]
display homepage()
new department creation successful+Homepage
[new department creation failed]
display homepage()
new department creation failed+Homepage
Actor
Viewer
Distributor
Administrator
www.websequencediagrams.com

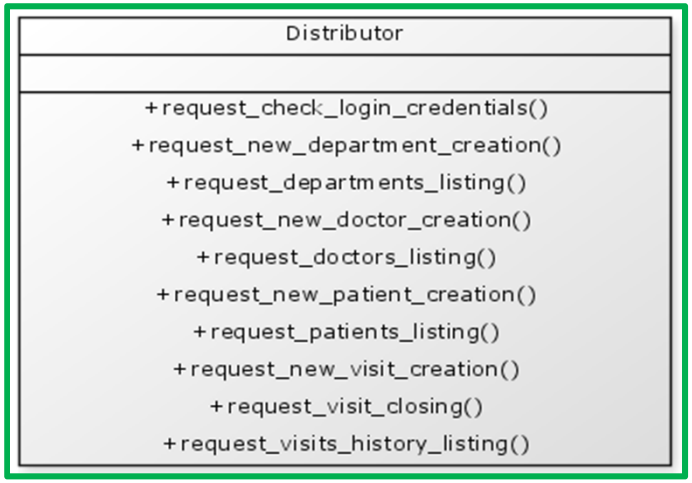

View departments list
Actor
Viewer
Distributor
Administrator
Click on menu item "View departments list"
new Distributor()
request departments listing()
new Administrator()
db list departments()
SQL select all department names
all department names
all department names
display department list page()
Departments list + Homepage
Actor
Viewer
Distributor
Administrator
www.websequencediagrams.com

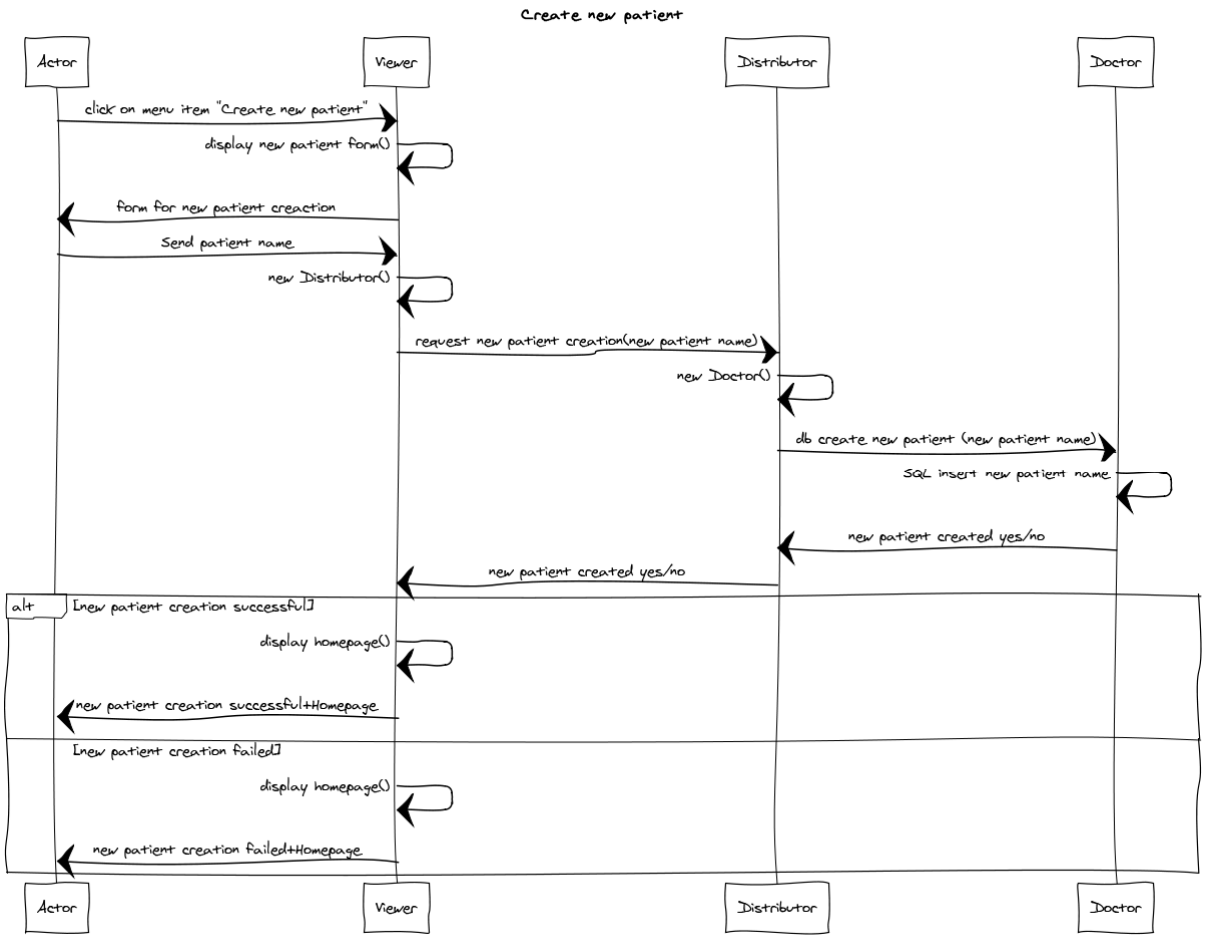

Create new doctor
Actor
Viewer
Distributor
Administrator
click on menu item "Create new doctor"
display new doctor form()
form for new doctor creation
Send doctor name
new Distributor()
request new doctor creation(new doctor name)
new Administrator()
db create new doctor (new doctor name)
SQL insert new doctor name
new doctor created yes/no
new doctor created yes/no
alt
[new doctor creation successful]
display homepage()
new doctor creation successful+Homepage
[new doctor creation failed]
display homepage()
new doctor creation failed+Homepage
Actor
Viewer
Distributor
Administrator
www.websequencediagrams.com

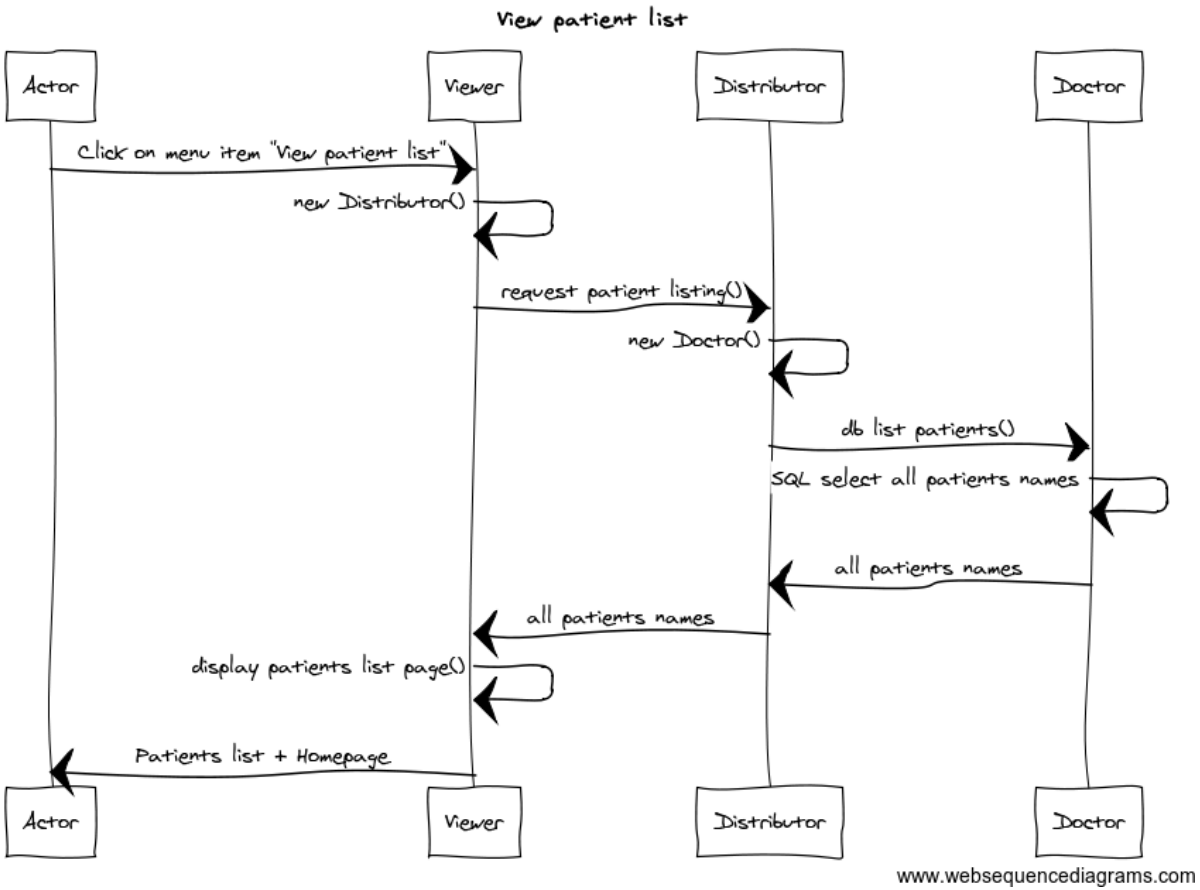

View doctor list
Actor
Viewer
Distributor
Administrator
Click on menu item "View doctor list"
new Distributor()
request doctors listing()
new Administrator()
db list doctors()
SQL select all doctors names
all doctors names
all doctors names
display doctors list page()
Doctors list + Homepage
Actor
Viewer
Distributor
Administrator
www.websequencediagrams.com

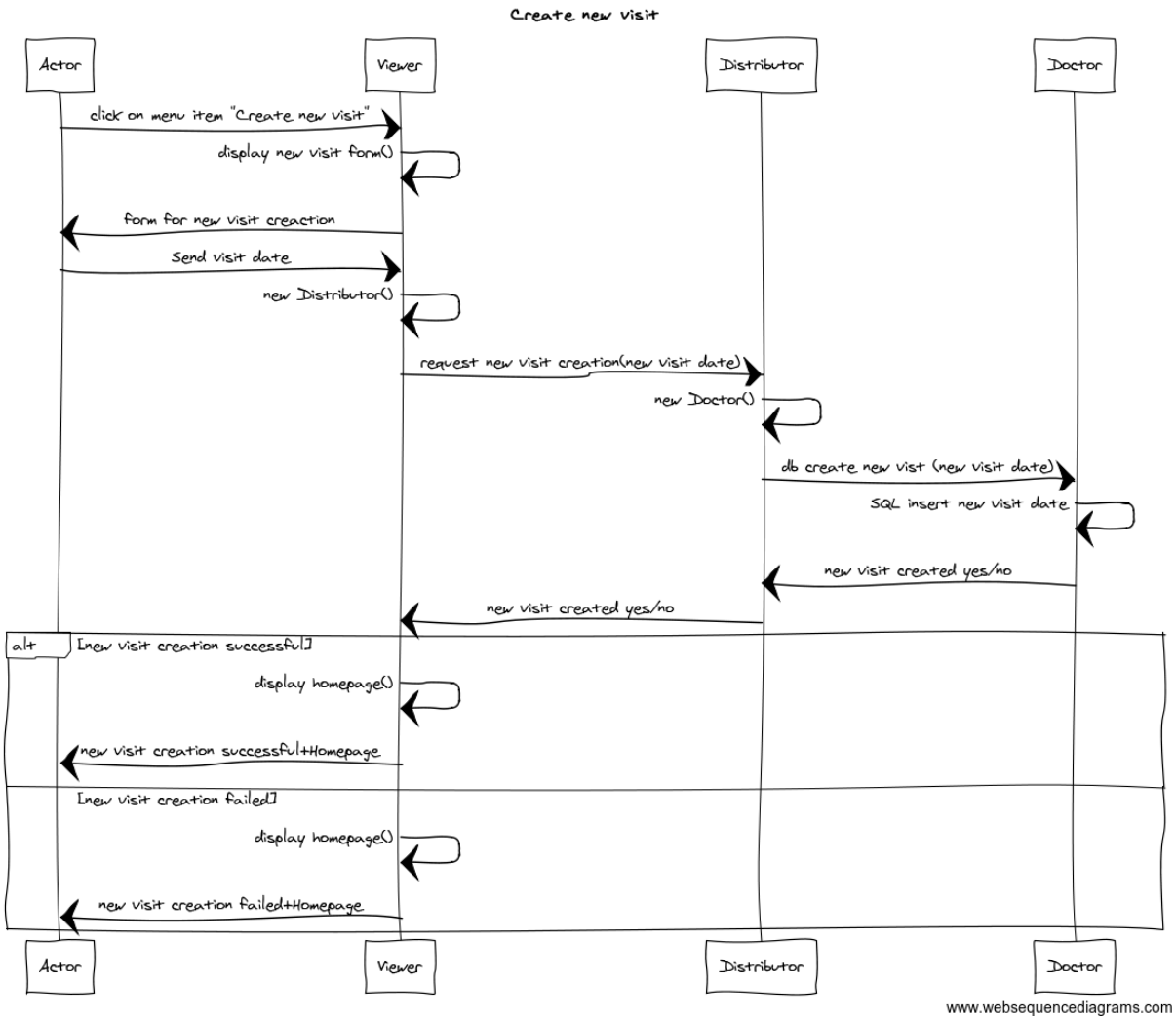
System design – application logic - doctor

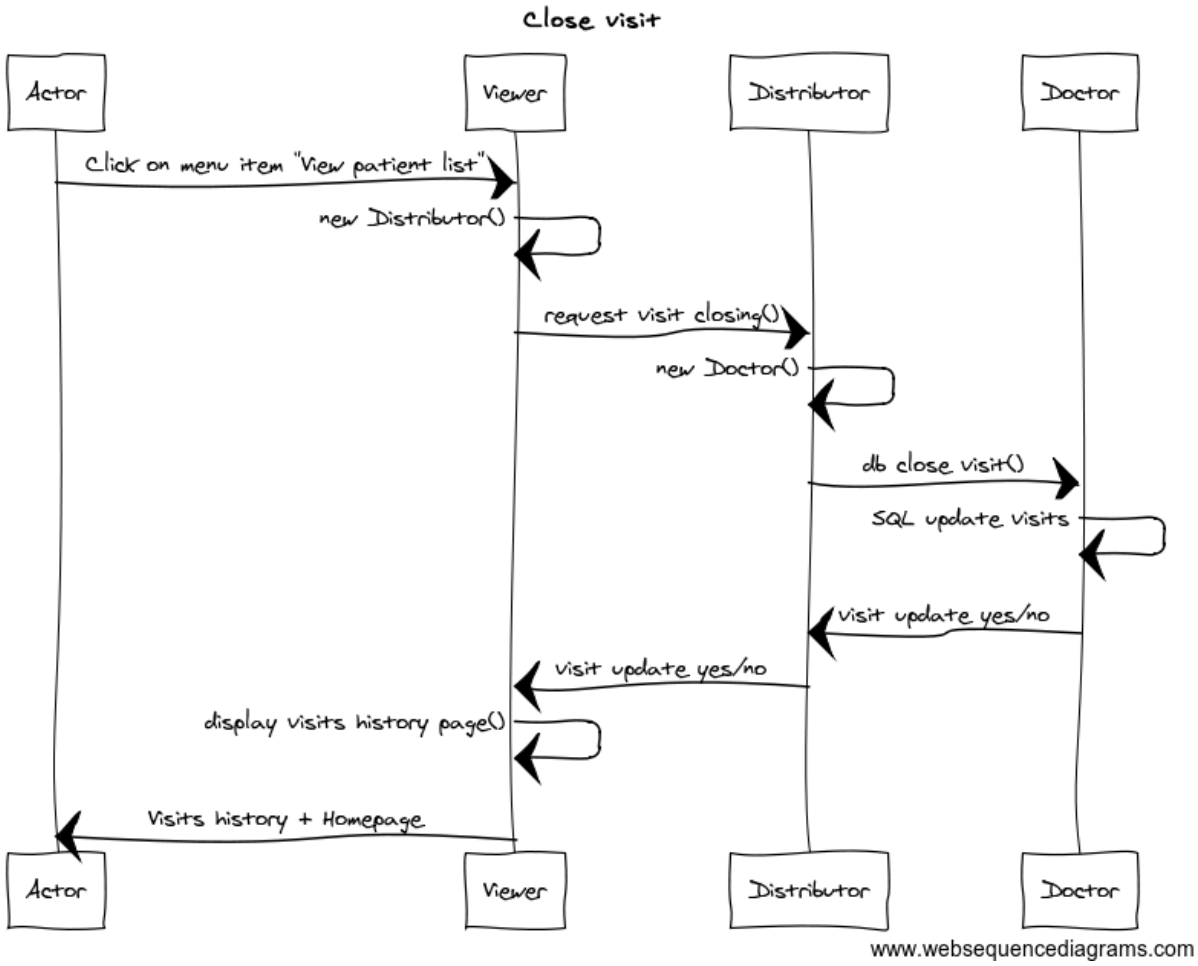
- **create a new patient**
  1. create an object from the class *GUI* and display the *patient creation form*
  2. get the patient name, age, birth number from the *patient creation form*
  3. create an object from the class *Doctor* and store the name, age, birth number for the new patient
  4. create an object from the class *GUI* and display the *patients listing page*
- **view list of patients**
  1. create an object from the class *Doctor* and fetch the list of patients
  2. create an object from the class *GUI* and display the *patients listing page*
- **create a new visit**
  1. create an object from the class *GUI* and display the *patients listing page*
  2. get the patient birth number of the selected patient from the *patients listing page*
  3. create an object from the class *Doctor* and store a new visit for the given patient
  4. create an object from the class *GUI* and display the *visits history page* for the given patient
- **view list of visits**
  1. create an object from the class *GUI* and display the *patients listing page*
  2. get the patient birth number of the selected patient from the *patients listing page*
  3. create an object from the class *Doctor* and fetch the list of visits for the given patient
  4. create an object from the class *GUI* and display the *visits history page* for the given patient
- **end a visit**
  1. create an object from the class *GUI* and display the *patients listing page*
  2. get the patient birth number of the selected patient from the *patients listing page*
  3. create an object from the class *Doctor* and update visit for the given patient
  4. create an object from the class *GUI* and display the *patients listing page*

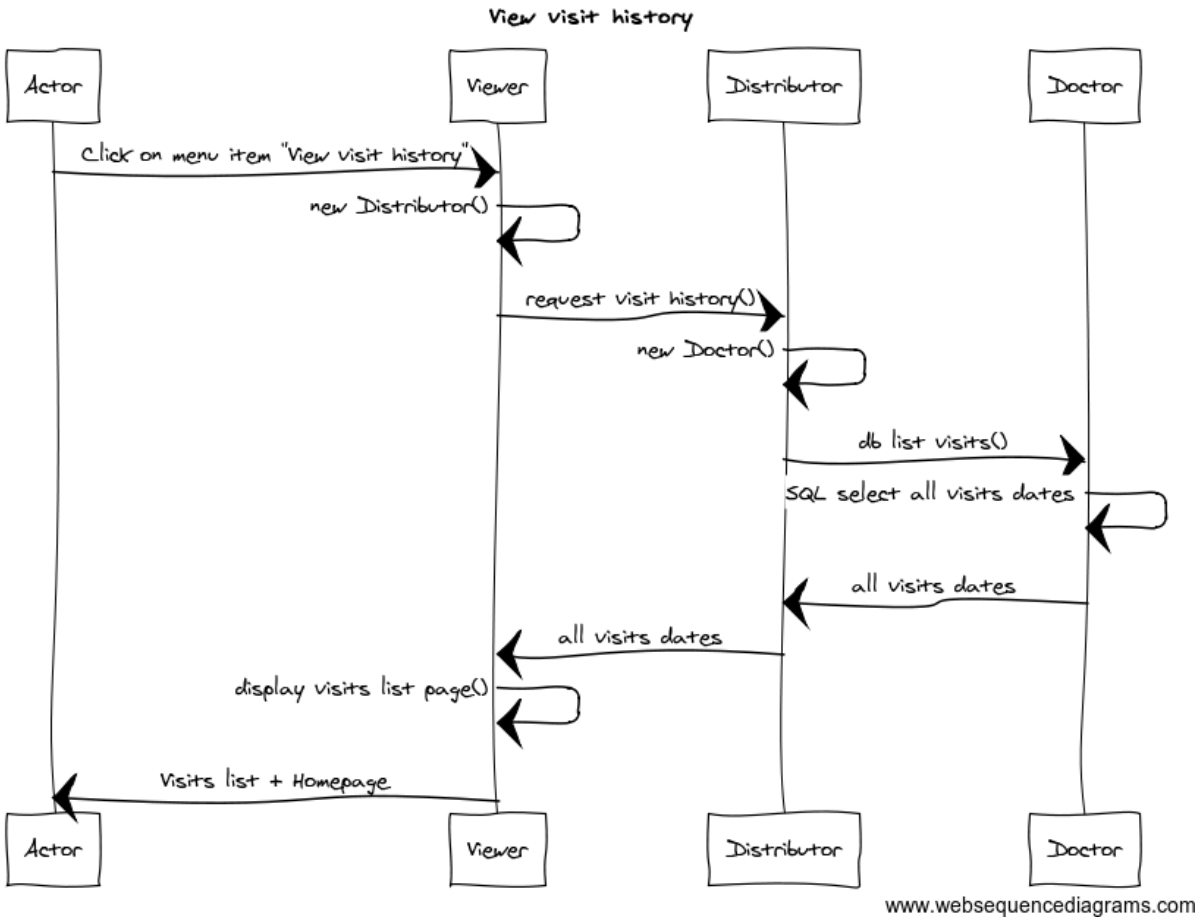


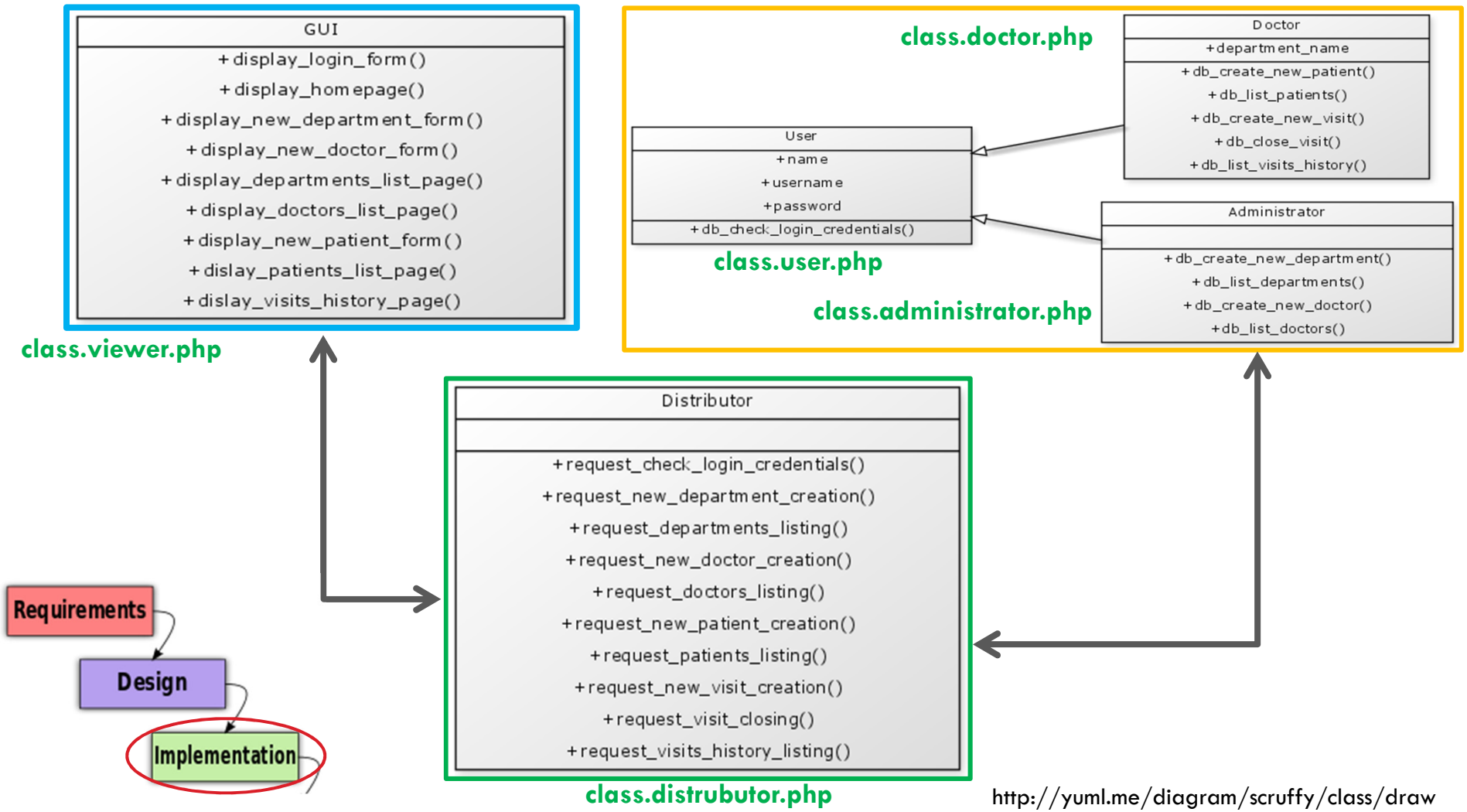

Create new patient
Actor
Viewer
Distributor
Doctor
click on menu item "Create new patient"
display new patient form()
form for new patient creation
Send patient name
new Distributor()
request new patient creation(new patient name)
new Doctor()
db create new patient (new patient name)
SQL insert new patient name
new patient created yes/no
new patient created yes/no
alt
[new patient creation successful]
display homepage()
new patient creation successful+Homepage
[new patient creation failed]
display homepage()
new patient creation failed+Homepage
Actor
Viewer
Distributor
Doctor
www.websequencediagrams.com


View patient list
Actor
Viewer
Distributor
Doctor
Click on menu item "View patient list"
new Distributor()
request patient listing()
new Doctor()
db list patients()
SQL select all patients names
all patients names
all patients names
display patients list page()
Patients list + Homepage
Actor
Viewer
Distributor
Doctor
www.websequencediagrams.com


Create new visit
Actor
Viewer
Distributor
Doctor
click on menu item "Create new visit"
display new visit form()
form for new visit creation
Send visit date
new Distributor()
request new visit creation(new visit date)
new Doctor()
db create new vist (new visit date)
SQL insert new visit date
new visit created yes/no
new visit created yes/no
alt
[new visit creation successful]
display homepage()
new visit creation successful+Homepage
[new visit creation failed]
display homepage()
new visit creation failed+Homepage
Actor
Viewer
Distributor
Doctor
www.websequencediagrams.com


Close visit
Actor
Viewer
Distributor
Doctor
Click on menu item "View patient list"
new Distributor()
request visit closing()
new Doctor()
db close visit()
SQL update visits
visit update yes/no
visit update yes/no
display visits history page()
Visits history + Homepage
Actor
Viewer
Distributor
Doctor
www.websequencediagrams.com

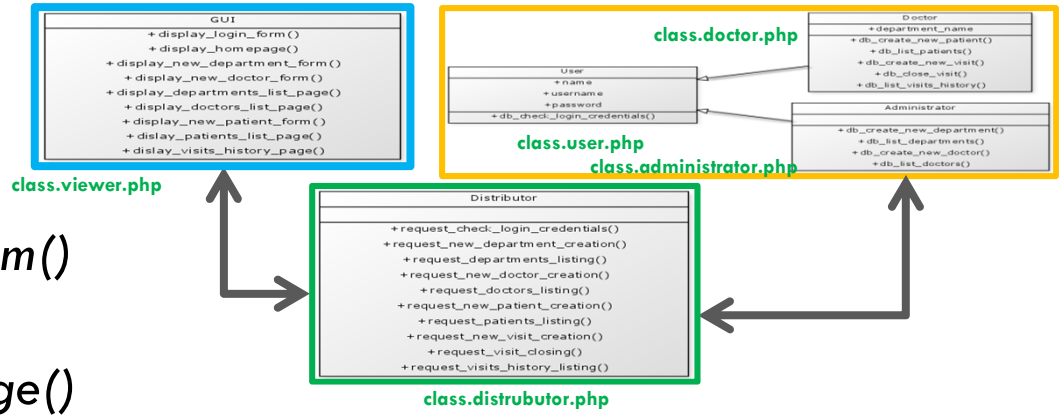

View visit history
Actor
Viewer
Distributor
Doctor
Click on menu item "View visit history"
new Distributor()
request visit history()
new Doctor()
db list visits()
SQL select all visits dates
all visits dates
all visits dates
display visits list page()
Visits list + Homepage
Actor
Viewer
Distributor
Doctor
www.websequencediagrams.com



http://yuml.me/diagram/scruffy/class/draw

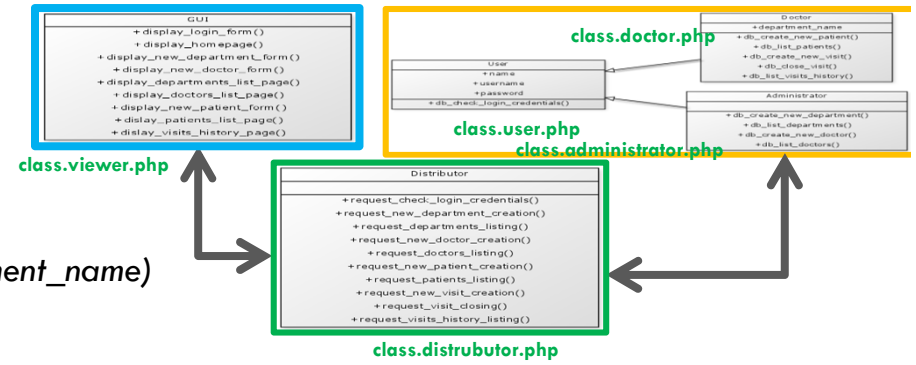
Implementation – Presentation tier

- Class for user interface: class.viewer.php
  - Operations
    - *display_login_form()*
    - *display_homepage()*
    - *display_new_department_form()*
    - *display_new_doctor_form()*
    - *display_departments_list_page()*
    - *display_doctors_list_page()*
    - *display_system_message()*
    - *display_new_patient_form()*
    - *display_patients_list_page()*
    - *display_patient_visits_page()*



Implementation – Logic tier

- Class for computation logic: **class.distributor.php**
  - Operations
    - *request_check_login_credentials($login, $password)*
    - *request_new_department_creation($department_name)*
    - *request_departments_listing()*
    - *request_new_doctor_creation($doctor_name, $department_name)*
    - *request_doctors_listing()*
    - *request_new_patient_registration($patient_name, $patient_age, $patient_birth_number)*
    - *request_patients_listing()*
    - *request_patient_receiving($patient_birth_number, $arrival_date)*
    - *request_patient_release($patient_birth_number, $departure_date)*
    - *request_patient_visits_listing($patient_birth_number)*



Implementation – Data tier

- Class for data persistence for all users: **class.user.php**
  - Operation
    - *db_check_login_credentials($login, $password)*
- Class for data persistence for administrative tasks: **class.administrator.php**
  - Operations
    - *db_create_new_department($department_name)*
    - *db_list_departments()*
    - *db_ create_new_doctor($doctor_name, $department_name)*
    - *db_ list_all_doctors()*
- Class for data persistence for clinical tasks: **class.doctor.php**
  - Operations
    - *db_ register_new_patient($patient_name, $patient_age, $patient_birth_number)*
    - *db_ list_all_patients()*
    - *db_ receive_patient($patient_birth_number, $arrival_date)*
    - *db_ release_patient($patient_birth_number, $departure_date)*
    - *db_ list_all_patient_visits($patient_birth_number*

Homework

- Deploy the hospital information system studied in the lecture to your Apache/PHP/MySQL installation.
  - Create a new database called **nis** in your MySQL system
  - Open a SQL console and execute the queries found in **database.sql**
  - Edit the **connection.php** and change the database user name ($databaseusername) and password ($databasepassword) if required
  - Copy the .php files to the web directory of your Apache installation
  - Test the application http://localhost/class.viewer.php?form_id=1
    - Login admin/admin or janda/janda

Syllabus of lectures and tutorials

		Lectures (45 min)	Tutorials (45 min)
Lesson 1	Sep 30	Medical Informatics and IS definition	OpenEMR
Lesson 2	Oct 7	HW infrastructure of IS	OpenEMR
Lesson 3	Oct 14	Operation systems	GaiaEHR
Lesson 4	Oct 21	Databases of IS	SQL
~~Lesson 5~~	~~Oct 28~~		
Lesson 6	Nov 4	Clinical oriented IS	SQL
Lesson 7	Nov 11	Decision support systems Medical data coding	OpenMRS
Lesson 8	Nov 18	Phase and IS development principles	UML
Lesson 9	Nov 25	Standard implementation methodology	Programing in HTML and PHP
Lesson 10	Dec 2	Standard implementation methodology	Programing in PHP and MySQL
Lesson 11	Dec 9	Data and communication standards	OpenMRS
Lesson 12	Dec 16	Presentation of practical project and final exam	

Plan for next week

- HTML, PHP