after every sprint 1 day for testing
everyone documents his or her own progress for themselves put it together in the wiki

Workpackages:

Before we start: set up github make a gantt chart

First sprint: 2 week (14 days)

1) Python/c# wfdb toolkit: 5 days - Michael

- how to implement the toolkit into C#

- get to know what is the input file format (example .mat , .dat. etc....) and output (example .jpg png. etc..)
- see if we can convert the received data into a unified data format which can be stored in the database

2) Login In Window design: 3 days - Laura

Design the Login windows already in WINDOWS FORMS-> 3 buttons for the different identification methods, design all the methods
NOT THE LOGIC!

3) Design of 4 different graphical user interfaces: 5 days - Mariana

WINDOWS FORMS
Every role gets its own GUI design. GUI is designed according to role. The different roles only see the buttons they are allowed to use. example -> nurse only has a button to see active patient

4) make a test plan: 3 days - Stefan

which components will be tested together - requirements for the test (when failed when not), customer requirements have to be fulfilled

MEETING after sprint - 1day

explaining progress to others and testing

Second Sprint: 3 weeks (21 days)

1) Database design and SQL database: 4 days - Michael

- ER diagram of the 3 databases
- set up a database and fill in dummy entries (local database with XAMP)

+ SQL connector: 1 day - Michael

is called separately from the main program logic at the start of the application

- there is a object of class user -> global variable of the user called “role” is assigned by the login logic to an integer - Stephan

- how to retrieve data from id/ e-card reader - Stephan/Laura

3) and 4) program logic: 16 days - log out - 1 day - Michael

2) Login Window logic: 6 days

- look at use case diagram what the program needs to be able to do:
read patient from GUI -> File search and file browser functions - 4 days
create user (admin!) - 1 day
create patient (read out data from ecard: when patients is created there should be a button that

says import data from ecard. When clicked the user should see a screen where he is prompted to put the ecard in the reader. After ecard data is received the appropriate data fields are filled out by themselves) - (GIN connection: read out the ecard data) - 7 days - down below

display ecg - 2 days - Michael

-which gui is loaded depending on the role variable of the user - 1 day - Stephan

getting data form usb do database: 1 day - Stephan

figuring out how to access the system file browser - easy!

putting everything together: 2 days

SUM of days: 46

Buttons:

Stephan **- Patient Info Button function:** distinguishes role of who is pressing

button

Hold the ecard to the reader and show the according patient Open patient info window.

Michael - **display ecg button:** calls the python script
(optional) **edit button (only for physician and admin):** Makes info editable

Roles: physician, nurse, patient, admin

Create Patient Button:
Stephan **- Auto-fill with e-card button:** fills in info from ecard
Stephan **- Add ECG-File button:** with file browser provide file path to ecg Michael **- Save Button:** Uploads file to the database

Roles: Admin, physician

Stephan **- Search Patient button:** opens the search window Role: physician, admin

Create User button: opens the create user window
Stephan - **Get ecard hex value button:** gets hex value of card Laura - **Get id card hex value button:** gets hex value of card Laura - **save button:** uploads to database

Role: physician, admin

Windows: Mariana + laura

Patient info window:

Display patient data , enter ecg viewing options, display ECG button, Button edit (only for physician and admin)

Search Window:

has text field where to search term, has a UI element where patient records are listed and can be clicked to open (button or double click depends on how the UI element works)

Create Patient Window:

Text fields to enter patient data, save Button, auto fill with ecard button, Add ECG-file button

Create User Window:

Text fields for input of data, two buttons for getting ecard/id-card hex values, save button

Patient information to be stored:

- patient identifier [int] (PK)
- name [string]
- surname [string]
- date of birth [string] (as defined by ecard) - social security number [int]

- gender/sex [boolean]
- medication [string]
- (optional) title
- ECG-files [.hea, .dat] or [BLOB]

User information to be stored in SQL (at least one log in option has to be stored)

- User id [int] (PK)
- Name [string]
- Surname [string]
- Username [string]
- Password [string]
- E-card number [hex] - id-card number [hex] - Role [int]

C# class user

- User id from SQL - Role

- Permitted [boolean]

notes:

deleting is not part of the admin user interface : ask sauermann?
logout button -> restarts the program
(optional) patient can look at their data without creating a user account u