OpenGeneMed: a portable, flexible and customizable informatics hub for the coordination of Next-Generation Sequencing studies in support of precision medicine trials.

Technical Documentation

Alida Palmisano, Ming-Chung Li, Eric Polley, Yingdong Zhao*, Richard Simon*
* (corresponding author)

Division of Cancer Treatment and Diagnosis (DCTD)
Biometric Research Program,
National Cancer Institute,
Rockville, MD, USA

Email inquiries to: alida.palmisano@nih.gov, zhaoy@ctep.nci.nih.gov

⚠This manuscript describes the default configuration of the system and does not reflect changes that your system administrator may have implemented in your version of OpenGeneMed. Before submitting any inquiry to the authors of this manuscript, please contact your system administrators to make sure your inquiry is not connected to changes they made to the default configuration.⚠

Version 1.0

December 2015

Contents

1 Document Introduction

This document is the technical documentation about the OpenGeneMed system.

Some of the text and figures is presented in a recent publication:

Zhao, Yingdong, Eric C. Polley, Ming-Chung Li, Chih-Jian Lih, Alida Palmisano, David J. Sims, Lawrence V. Rubinstein et al. "GeneMed: An Informatics Hub for the Coordination of Next-Generation Sequencing Studies that Support Precision Oncology Clinical Trials." Cancer informatics 14, no. Suppl 2 (2015): 45.

This document contains a detailed explanation about how OpenGeneMed is implemented and how to customize the code for specific study needs. A different document (called 'User Manual') contains step by step instructions about how to use the default configuration of OpenGeneMed.

⚠This manuscript describes the default configuration of the system and does not reflect changes that your system administrator may have implemented in your version of OpenGeneMed. Before submitting any inquiry to the authors of this manuscript, please contact your system administrators to make sure your inquiry is not connected to changes they made to the default configuration.⚠

1.1 Abstract

We developed an informatics system, called OpenGeneMed, to support clinical trial research.

The system is an open and customizable version of the GeneMed system (Zhao et al. (2015)), a web-based interface developed for the National Cancer Institute (NCI) Molecular Profiling based Assignment of Cancer Therapy (MPACT) clinical trial (NCT01827384) conducted in the National Institutes of Health (NIH) Clinical Center.

Similar to the GeneMed system, OpenGeneMed streamlines the work flow of the clinical trial and it can be used by clinicians, lab personnel, statisticians and other researches as a communication hub. The web system automates the annotation of the genomic variants identified by sequencing, classifies the actionable mutations according to customizable rules, and facilitates quality control by the molecular characterization lab in the review of variants.

OpenGeneMed collects baseline information about the patients from the clinic team to determine eligibility for the panel of drugs available. The system generates patient reports containing detected genomic alterations providing summarized information that can be used for treatment assignment by a supervising treatment review team.

OpenGeneMed is distributed as a standalone virtual machine, ready to be deployed and used from a web browser. The code of the informatics system is written in a modular way: this allows easy customizion of the existing features and addition new modules to address the needs of specific clinical trials and research teams. Several examples on how to customize the code are provided in the technical manual distributed with the virtual machine.

In summary, OpenGeneMed offers an initial set of features inspired by our experience with GeneMed, a system that has been proven to be an efficient and successful informatics hub for coordinating the reliable application of next-generation sequencing in the MPACT trial. OpenGeneMed facilitates the exchange of real-time information used by the medical and research staff for therapeutic decision-making. A key additional capability of OpenGeneMed is that it is open-source and offers easy customization and expansion.

2 Introduction to the OpenGeneMed system

Next-generation sequencing (NGS) technologies have been shown to be powerful tools for the clinical diagnosis and treatment of patients with cancer and other diseases, due to the increase of sequencing speed, reduction of cost, and improvement in accuracy over the past decade (Rothberg et al. (2011)). Precision medicine treatments for cancer are gaining momentum due to increase interest in approaches guided by next-generation sequencing (Doroshow (2010); Gargis et al. (2012); Simon and Polley (2013); Simon and Roychowdhury (2013); Tran et al. (2013); Vogelstein et al. (2013)).

The National Cancer Institute (NCI) is now conducting a clinical trial (NCT01827384) titled MPACT (Molecular Profiling based Assignment of Cancer Therapy) to treat late stage cancer patients based on their targeted gene sequencing profile (Kummar et al. (2015)). Patients with progressive solid tumors who have exhausted standard treatments undergo biopsy and have their tumors characterized based on amplicon sequencing of over 380 unique actionable variants in 20 genes.

In MPACT several treatment regimens are used and the sequencing assay is designed to identify mutations likely to de-regulate some specific pathways: for details about the trial we refer to (Kummar et al. (2015)). Mutations are called actionable if they fulfill some pre-specified rules. The rules for actionability were derived from literature documentation, reporting in the Catalogue of Somatic Mutations in Cancer (COSMIC) database (Bamford et al. (2004)) and evidence of functional effects for loss-of-function genes. Patients whose tumors contain at least one actionable mutation are randomized to either the actionable study treatment prospectively identified to work on that mutation/pathway or a control arm consisting of one of the other treatments. The informatics support needed to coordinate the complex set of rules and operations of this clinical trial motivated the creation of the informatics system called GeneMed (Zhao et al. (2015)). Similar principles has been used to develop our open and customizable version of the system called OpenGeneMed.

A key feature of the informatics system is to serve as a communication hub linking the different teams involved in daily management of a clinical trial (sequencing lab, treatment review team, clinic team, statisticians, researchers, etc.). In addition to the bioinformatics pipeline for raw data processing and sequence alignment, the system must have the ability to quickly annotate the sequencing data, predict the functional impact of mutations, and match the detected mutation to the pre-identified genomic changes in tumors found in previous studies (i.e., the actionable mutations of interest). The informatics hub should have the ability to translate and present the mutation findings from a sequencing results into a concise and easy-to-read actionable report for the treatment review team in a timely manner. This information will then be used by the team to assign patients to different arms based on their detected mutations. Clinical trials follow a variety of strategy to assign patients to treatment arms, one of the most commonly used being randomization. While the GeneMed system includes the assignment of the patient to the proper treatment based on randomization tables, OpenGeneMed does not include a module to do this assignment because we wanted the system to be useful for all kind of trials, wheter randomized or not. However the architecture of the system has been designed to allow easy integration of additional modules (some examples presented in this document, with more details available in the technical documentation): in this way the addition of an automated (or semi-automated) treatment assignment module should be an easy task for the IT group to set up the informatics system.

OpenGeneMed contains a module to collect and store patient data, including response data for prior therapies: this data is essential to evaluate eligibility for treatments in the study, assisting the treatment review team and the clinic team in the development of a treatment plan. The system is also equipped to make both sequencing data and clinical response data available to statisticians who are able to evaluate whether there is any improvement in response rate or progression free survival (PFS) favoring assignment of therapy based on genetic sequencing and to determine which actionable mutation-treatment pairs lead to improved patient outcome. OpenGeneMed contains an administrative module that allows the system manager to register users with different roles (e.g., Clinical Team, Lab Team, Treatment Review Team, etc.). Each group will be granted different permissions on data monitoring, uploads and changes. This module, as all others in OpenGeneMed, is customizable and extendable so that each clinical trial setup and workflow can be easily mapped into the system.

3 Implementation of the OpenGeneMed system

The OpenGeneMed system has been designed and developed by the Biometric Research Branch in the Division of Cancer Treatment and Diagnosis of the National Cancer Institute.

The OpenGeneMed system is operating on Virtual Machine running a CentOS 64 bit Linux (CentOS 6.7, kernel 2.6.32) and it uses MySQL (version 5.1.73) as back-end database. Technical documentation about how to setup and initialize the system is provided in the following sections.

The application is built on a client-server architecture, which provides multiuser access to the application. The application can be launched in any modern web browser and presents an intuitive graphic user interface.

The front end (client side) interface is scripted using HTML, CSS, and JavaScript. The back end (server side) are scripted using Perl (details on modules used are provided below). Java (version 1.8.0_51) is installed as pre-requirement to run the annotation software snpEff (version 3.4, released Jan 16, 2014, downloaded from http://sourceforge.net/projects/snpeff/files/).

There are 21 MySQL tables in the default setup of the OpenGeneMed system. Details of the database tables are listed in Section 3.7 on page 10.

The rest of this chapter focuses on technical details about how to setup the system, access the client and customize the code. If you are interested in learning how to use the exiting system and to see screenshots of OpenGeneMed in action, please refer to the User Manual.

3.1 Initial server setup

The server runs on a Virtual Machine (CentOS 64-bit). An external software is required to run the virtual machine: we tested our system with Virtual Box (version 4.3.12, host OS: Windows 7 Professional, 64-bit). It is possible to run our 64-bit virtual machine on a 32-bit host if the hardware allows it, however we do not recommend it. Please refer to VB user manual for more details.

To transfer file from/to the machine, you can use a network file transfer application (e.g., PuTTy) or take advantage of the 'Shared folder' capabilities of Virtual Box. Please refer to VB user manual for more details.

To access the VM, use `opengenemed` as user and `123opengm` as password. The `opengenemed` is has sudo privileges. The password for the root user is `123opengmroot`. We recommend to change these values using common unix commands.

The 'opengenemed' process should be already running as it is listed in the processes loaded at startup.

To check its status, stop it, restart it, use the following console command:

```
sudo /etc/init.d/opengenemed [start|restart|stop|status]
```

The mysql database information is listed below. We recommend to change this password as well. Make sure to change the password both within mysql and in OpenGeneMed files that use this information (/opt/applications/opengenemed/webroot/script/opengenemed.pl and /opt/applications/opengenemed/config/local/settings.ini).

```
mysql -u root -D opengenemed -p

password: 123opengmdb
```

3.2 Network setup

There are a few different options to allow users connect to the VM through the network.

1. Local connection on the host machine.
   This allows to access the VM and OpenGeneMed from a browser on the same machine running Virtual Box. This setup is a good first option to troubleshoot network setup, but it is also useful for uploading and annotating vcf files locally as well as querying the data locally.

   After downloading the .ova file of the VM, the user need to set up a 'Host-only' connection using Virtual Box options. This allow the communication between the VM and the host machine. Note that the instructions below assume that the user doesn't already have a Host-only connection. If a connection already exist, the user can modify the IP addresses as explained below or add a new adapter.

   To set up the connection, in Virtual Box, select File/Preferences/Network/Host-only network. Add a single adapter: ipv4 address 192.168.56.1, ipv4 mask 255.255.255.0, DHCP Server, Enable Server, address: 192.168.56.100, mask 255.255.255.0, lower bound 192.168.56.101, upper bound 192.168.56.254. These IP address are given as example and users can change them as desired, but be aware that all the instructions that follow use the above addresses. In Virtual Box, right click on the VM/Settigs. Disable Audio. Network, Adapter 1, Enable Adapter, NAT (this only to grant the VM access to the web for updates and other operations). Adapter 2, Enable Adapter, Host-only, Name: [the name of the adapter created above].
   Start the VM, open a browser in the host machine and access the following address: **http://192.168.56.103**. If you see the login page of OpenGeneMed, everything is setup correctly.
   If not, you may have to access the VM, and through the centos console, make sure that the file `/etc/sysconfig/network-scripts/ifcfg-eth1`, contains DEVICE='eth1' BOOTPROTO='dhcp' ONBOOT='yes'; IPADDR=" NM_CONTROLLED=yes. If you make any change, you may need to restart the network service with `service network restart`. 'ifconfig' should show the ip of the machine (192.168.56.xxx). If not, you may have to change a different ifcfg-eth* file or some other configurations are not correct.
   If you are able to connect to the VM from the host via PuTTY, WinSCP or similar program, you need to make sure apache is correctly configured to run OpenGeneMed.
   For example, if we want the server to be available at 192.168.56.103, we will have to make the following changes to the following files:
   `sudo vi /etc/httpd/conf/http.conf` (change lines, in appropriate sections: 'Listen 192.168.56.103:80', 'ServerName 192.168.56.103:80')
   `sudo vi /etc/hosts` (add line '192.168.56.103 opengenemed.v1 opengenemed')
   `sudo service httpd start`
   `sudo chkconfig --level 345 httpd on`
   To check if the server works: `curl -s http://192.168.56.103 | more` (if text is shown, the server is up)
   From host, use a web browser to connect to http://192.168.56.103. If it doesn not work do the following:
   `su`, `service iptables stop` test now from host, it should work. If it does, make the change permanent with the following: (as root) `chkconfig --level 345 iptables off`

2. Set up connection within a company network: port forwarding.

   This allows to access the VM and OpenGeneMed from a browser on a machine different from the one running Virtual Box. This setup is a good in a situation where the different groups that need to interact with OpenGeneMed are on the same company/group network. A similar setup could be used to open the connection to the public, but additional instructions depending on the user company/group rules may apply and we cannot give instructions about those special cases.

   ⚠ Even if the final goal is to use this setup, please always follow the instructions of the "local connection" first. The additional steps to allow the connection from other machines are listed assuming that your Host machine is a Linux OS (instructions for a Windows host machine are below).

On the Linux Host machine (from the console, note that you need administrative privileges):

```
VBoxManage modifyvm "OpenGeneMed" --natpf1 "guestssh,tcp,,8787,192.168.56.103,80"
sudo iptables -A INPUT -p tcp --dport 8787 -j ACCEPT
sudo sh -c '/sbin/iptables-save > /etc/iptables.save'
sudo iptables-restore < /etc/iptables.save
```

In /etc/rc.local add the following line: `/sbin/iptables-restore < /etc/iptables_rules`

At this point, Virtual Box should redirect all the traffic coming to the Host IP address (port 8787) to the VM (OpenGeneMed). To identify the IP address of the Host machine, use the `ifconfig` command. Now any user in the company/group network should be able to access OpenGeneMedfrom the browser at `http://[Host_IP_address]:8787`.

On a Windows machine (from the console, notat that you need administrative privileges), navigate to the installation directory of Virtual Box (usually Program Files/Oracle/VirtualBox). and run the command `VBoxManage modifyvm "OpenGeneMed" --natpf1 "guestssh,tcp,,8787,192.168.56.103,80"`. Next time you access Virtual Box, Windows may ask you to set up the Firewall to allow VB to accept traffic on the newly opened port. You should setup the inbound rules for this connection through the Windows Firewall options.

Open Windows Firewall by clicking the Start button/Control Panel/Windows Firewall. In the left pane, click Advanced settings (Administrator permission required). If you're prompted for an administrator password or confirmation, type the password or provide confirmation. In the Advanced Security dialog box, in the left pane, click Inbound Rules, and then, in the right pane, click New Rule. Follow the instructions in the New Inbound Rule wizard to allow incoming traffic from port 8787. If youâĂŹre having trouble allowing other computers to communicate with your computer through Windows Firewall, you can try using the Incoming Connections troubleshooter to automatically find and fix some common problems.

⚠ Please be aware that network connection problems are usually not connected to OpenGeneMed as they may be due to incorrect adapters setup, firewalls, anti-virus programs and other network policies from your company/group network. If you followed carefully the above instructions and you still encounter problems, we encourage you to start by getting in touch with your network administrators. If they cannot help, you may contact the authors of this manuscript, but be aware that we may not be able to help on issues beyond OpenGeneMed.

3.3 Email delivery setup

OpenGeneMed is designed to allow automatic delivery of email messages to people that need to complete specific actions within the system (e.g., reviewing variants for a sample, assigning a treatment to a patient, delinking a study participant, etc.).

Since the email delivery requires the Virtual Machine to be setup according to your network specifications, this feature is disabled in the initial setup of OpenGeneMed.

To turn it on, just few entries in a configuration files need to be changed (see below), however before making the changes, please test the commands below to make sure the VM is able to send emails through the 'sendmail' command.

```
cat >/tmp/email.txt
Subject: test email
text of test email
[Ctrl-D]
sendmail [email_address] </tmp/email.txt
```

If the email is not delivered, the sendmail program is not setup correctly.

You may need to create a file called 'relay_domains' in the /etc/mail directory with the proper IP address of destination domains (and subdomains thereof) the VM will relay mail to. You may also need to change the /etc/mail/sendmail.cf file to use the proper 'Smart' relay host. You may also need to put the same information in the /etc/postfix/main.cf under the 'Internet or intranet' section.

⚠ All these configurations depends on the network where the VM is going to be used, so please refer to your network administrator for help and support.⚠

If the email is delivered correctly, you can modify the `system_mail` section in the file '/opt/applications/opengenemed/config/local/settings.ini' as follows:

- `send_email_enabled = 1` (0 = disabled, 1 = enabled)
- `from = OpenGeneMed@example.com` (this will be the "from" address used by all OpenGeneMed's automatically generated emails)
- `all_emails_to =` (leave empty on production servers, this can be used in **testing/debug settings only** to redirect all emails to a specific email address. More than one address can be used. Multiple addresses should be listed as a comma separated sequence)

Changes to the 'settings.ini' file will be effective only after opengenemed service is stop/restarted.

3.4 Database clean up

The distributed version of OpenGeneMed comes with a pre-populated database that contains information of few mock patients, with simulated samples, treatment assignments, eligibility criteria, etc. This is done to allow first-time users and reviewers to navigate a fully functional system and evaluate OpenGeneMed capabilities in its default configuration.

However it is important to clean up the database to prepare the system for deployment in real case studies. To make this operation quick and easy, we provided a script that will empty the appropriate tables and initialize the content of the database.

```
mysql -u root -p -D opengenemed < /opt/applications/opengenemed/config/local/CLEAN_databasedump.sql
```

⚠ It is very important to use this clean database dump file instead of manually emptying tables in the database, as the content of some tables is used to grant access to the system and to provide back end support for the annotation steps. Manual changes to the database tables' structure and content should be done with great care as they will affect OpenGeneMed operations in major ways. ⚠

3.5 Accessing the client

Once the VM is running, type the IP address of the VM in any web browser to use the OpenGeneMed system. If you don't see a login screen there may be something wrong with the VM setup: contact the IT personnel responsible of this setup for any help on these issues (the server may be running behind a firewall, or the VM has not be configured correctly, or the VM does not interface correctly with the network you are trying to access).

If the VM is setup correctly, the login screen should be visible and the IT personnel that setup the VM should have provided you details on which username and password to use to access OpenGeneMed. In the basic initial setup, eight accounts are added for demo purposes (usernames and passwords in Table 1). However the first recommendation while setting up the VM is to change/delete these accounts, so they may not work. Please contact your IT team with questions about this.

Role	Username	Password
Admin	opengenemed_admin@example.com	aaaAAA111
Clinic Team	opengenemed_clinicteam@example.com	bbbBBB222
Lab Team	opengenemed_labteam@example.com	cccCCC333
Lab Team Chief	opengenemed_labteamchief@example.com	dddDDD444
Clinic Team Chief	opengenemed_clinicteamchief@example.com	eeeEEE555
Study Coordinator	opengenemed_studycoordinator@example.com	fffFFF666
Statistician	opengenemed_statistician@example.com	gggGGG777
Treatment Review Team	opengenemed_treatmentteam@example.com	hhhHHH888

Table 1: *Eight accounts added in the OpenGeneMed system for demo purposes. The first recommendation while setting up the VM is to change/delete these accounts, so they may not work on your specific system: please contact the IT team that set up your system with questions about this issue.*

Details about how to use the different features of the system are presented in the User Manual.

3.6 High level description of the code

The back-end code for the system is mainly written using Perl. The main directory for the project is '/opt/applications/opengenemed/webroot'. Under this main folder, important subfolders are:

- lib: contains all the Perl modules used to run OpenGeneMed
- templates: contains the pieces of code needed by Mojolicious (see below) to build the front end web pages displayed in the browser
- script: contains the main script that starts the Mojolicious application
- log: contains log files that can be used to monitor the system usage and/or to debug the application
- public: contains javascript and css used for the management of the web pages. It also contains subdirectories that stores the files uploaded in the system for processing (vcf)

OpenGeneMed is written using Perl (with DBIX, and Mojolicious as the main libraries), Javascript, CSS and D3 (for the web interface). In order to customize the code, developers need to get familiar with these languages and libraries as well as MySQL (used for the back-end database management).

The system uses the morbo (developer server) of Mojolicious and it may not be good for production servers. It is used in this release because its autorestart features are exactly what makes it easy and fast to customize. For more information about deployment of Mojolicious applications, please refere to http://mojolicio.us/perldoc/Mojolicious/Guides/Cookbook#DEPLOYMENT.

The following sections describe in details the structure of the database and the different files that are used by the system in its default deployment. Section 3.9 on page 20 illustrates some examples of possible customization to the code to adapt OpenGeneMed to specific study's needs.

3.7 Database structure

The database (called opengenemed) contains 21 tables.

Eight of these tables are non-empty as they contain data used to build essential parts of the systems (drop downs with choices like the system user roles, or information used by the variant calling software to perform the calls). The content of most of those tables cannot be modified through the OpenGeneMed interface and we highly discourage manual changes done directly through mysql. However changes to both content and structure of some tables may be needed in order to address specific study needs. We refer the developer to Section 3.9 on page 20 for examples and guidelines about how to perform this task.

The remaining 12 tables store data related to patients, treatment, variants, users, etc. and they are empty in the initial setup of OpenGeneMed.

Below a detailed account of tables structure and content.

Table name (number of entries)	Field	Type
actionable_rule (1)	id	integer
	display_name	varchar(128)
	reason	varchar(255)
	date_created	datetime
	date_modified	timestamp
	date_deleted	datetime
actionable_rule_clause (1)	id	integer
	actionable_rule_id	integer
	actionable_rule_field_id	integer
	clause_type	enum ('include','exclude')
	operator	enum ("=","!=","<",">","<=",">=")
	filter_value	varchar(128)
	date_created	datetime
	date_modified	timestamp
actionable_rule_field (10)	id	integer
	table_name	varchar(64)
	column_name	varchar(64)
	display_name	varchar(128)
	date_created	datetime
	date_modified	timestamp
	is_numeric	tinyint
biopsy (0)	bid	integer
	pid	varchar(20)
	biopsy_site	varchar(30)
	biopsy_phase	varchar(20)
	date_biopsy	varchar(10)
	drug_biopsy	varchar(20)
	date_submit	datetime
	submitter	varchar(20)
	comment	varchar(200)
	biopsy_type	varchar(20)

cosmic (5706)	gene	varchar(20)
	cosmicid	integer
	mcds	varchar(50)
	alt	varchar(20)
	maa	varchar(50)
	mdec	varchar(50)
	mgp	varchar(100)
	mgpstrand	varchar(10)
	chrpos	varchar(50)
eligibility (0)	qid	integer
	pid	varchar(20)
	answer	varchar(50)
	submitter	varchar(20)
	date_created	datetime
	date_modified	timestamp
eligibility_questions (0)	qid	integer
	answer	varchar(50)
	submitter	varchar(20)
	note	varchar(100)
	date_created	datetime
	date_modified	timestamp
	date_removed	datetime
gpd (22)	id	integer
	gsymbol	varchar(20)
	pathway	varchar(20)
	pathway	varchar(100)
	gene_type	varchar(20)
	moitype	varchar(20)
	pathwaylink	varchar(200)
	date_modified	timestamp
	date_removed	datetime
moi_current (13)	moi_id	integer
	chr	varchar(10)
	start	integer
	cosmicid	integer
	ref	varchar(200)
	alt	varchar(200)
	chrpos	varchar(50)
	actionable	enum ["Yes", "No"]
	reason	varchar(50)
	cnt	integer
	gene	varchar(10)
moi_current_log (0)	moi_id	integer
	id	int(5)
	event	varchar(40)
	submitter	varchar(20)
	date	timestamp
patient (0)	pid	varchar(20)
	sex	varchar(1)
	dx	varchar(20)

	dx_date	varchar(10)
	age_dx	integer
	response	varchar(50)
	duration	decimal(3,1)
	date_of_seq	date
	date_of_mutsummary	date
	dx_subtype	varchar(50)
	date_submit	varchar(30)
	submitter	varchar(20)
	date_updated	timestamp
	comments	varchar(500)
	randomid	varchar(20)
polymer (4613)	chr	varchar(3)
	start	integer
	end	integer
	width	integer
	strand	varchar(1)
	ampliconId	varchar(20)
prior_therapy (0)	ptid	integer
	pid	varchar(20)
	regimen	varchar(50)
	date_rx	varchar(10)
	best_response	varchar(20)
	duration	varchar(10)
	date_submit	varchar(30)
	submitter	varchar(20)
	comment	varchar(200)
sample (0)	sid	integer
	pid	varchar(20)
	date_seq_submit	varchar(20)
	submitter	varchar(20)
	review_status	varchar(11)
	review_date	varchar(30)
	date_sample	varchar(10)
	autosid	integer
snpindel (2129)	id	integer
	chr	varchar(3)
	start	integer
	dbsnpid	varchar(20)
	ref	varchar(200)
	alt	varchar(200)
	maf1	decimal(4,2)
	maf2	decimal(4,2)
	maf3	decimal(4,2)
system_user (8)	id	integer
	username	varchar(128)
	password	varchar(64)
	date_created	datetime
	date_modified	timestamp
	first_name	varchar(256)
	last_name	varchar(256)

	system_user_role_id	int(10)
system_user_role (8)	id	integer
	date_modified	timestamp
	display_name	varchar(64)
todo_task (0)	id	integer
	date_req	datetime
	submitter	varchar(20)
	req_type	varchar(100)
	pid	varchar(20)
	recipient	varchar(20)
	date_closed	timestamp
treatment_assignment (0)	tid	integer
	pid	varchar(20)
	treatment	varchar(30)
	arm	varchar(30)
	submit_date	timestamp
	submitter	varchar(20)
	arm	varchar(20)
	comments	varchar(200)
	ack_by	varchar(20)
	ack_date	timestamp
treatment_report (0)	tid	integer
	pid	varchar(20)
	regimen	varchar(20)
	date_rx	varchar(10)
	best_response	varchar(20)
	ncycles	integer
	date_prog	varchar(10)
	reason	varchar(200)
	date_submit	varchar(30)
	submitter	varchar(20)
	comment	varchar(200)
variant (0)	vid	integer
	pid	varchar(30)
	sid	varchar(30)
	chr	varchar(10)
	start	integer
	ref	varchar(100)
	alt	varchar(100)
	af	decimal(4,2)
	dp	integer
	gsymbol	varchar(50)
	gid	varchar(50)
	ncodon	varchar(50)
	pcodon	varchar(50)
	timpact	varchar(50)
	sift	varchar(50)
	siftcode	varchar(50)
	dbsnpid	varchar(20)
	chrpos	varchar(50)
	aid	varchar(30)

	moi	varchar(3)
	cosmicid	varchar(100)
	pathway	varchar(100)
	drug	varchar(100)
	nocallreason	varchar(100)
	af2	decimal(4,2)
	dp2	integer
	polymer	varchar(3)
	amoi	varchar(10)
	moitype	varchar(20)
	review	varchar(1)
	comment	varchar(500)
	reviewer	varchar(20)
	review_date	varchar(30)
	reason	varchar(50)
	cnt	varchar(30)
	maf1	decimal(4,2)
	maf2	decimal(4,2)
	maf3	decimal(4,2)

3.8 File system structure

OpenGeneMed's different components need to be stored in specific locations of the file system. The table below lists the major directories with their relevant content and usage. Most of the Perl object have a direct correspondent in a tab of the user interface and the name assigned to the file should clarify the relationship. Directories are listed in no particular order.

Directory	Content (Files/Subdirs)	Notes
/opt/applications/opengenemed/	config	configuration and initialization files
	webroot	webserver main directory
	docs	copy of user manual and technical documentation
	libs	perl and mojolicious libraries
/opt/applications/opengenemed/config/local	settings.ini ⚠	directories and other configuration settings. Changes to this file will be effective only after opengenemed service is stop/restarted.
	keyword.txt	used to annotate the variants. For details, refer to the code in AnnotateVCF.pm
	CLEAN_databasedump.sql	initial setup of tables and content as in the default configuration
	clear_tables.txt	sql directives to empty all the tables that are not prepopulated with default configuration
/opt/applications/opengenemed/webroot	lib	all mojolicious and dbix classes
	log	log files
	public	css, js, images and other support files. The main upload directory is also here.

	script	contains the script starting the main application
	templates	all mojolicious templates to control web GUI
/opt/applications/opengenemed/webroot/lib	OpenGeneMed	application classes
	Schema	DBIx classes
	Util	utility classes
	OpenGeneMed.pm	main application class
/opt/applications/opengenemed/webroot/lib/Util	Caller.pm	utility file to determine the method called before a failure (for debugging)
	Die.pm	utility file to manage failures at different stages
	Timer.pm	utility file to track execution time of different operation (for debugging and profiling)
/opt/applications/opengenemed/webroot/lib/Schema	Profiler.pm, ResultSet.pm, ResultSetBase.pm	DBIx utility classes
	Result	a class for each table in the db
	Result/ActionableRule.pm	
	Result/ActionableRuleClause.pm	
	Result/ActionableRuleField.pm	
	Result/Biopsy.pm	
	Result/Cosmic.pm	
	Result/Eligibility.pm	
	Result/EligibilityQuestions.pm	
	Result/Gpd.pm	
	Result/MoiCurrent.pm	
	Result/MoiCurrentLog.pm	
	Result/Patient.pm	
	Result/Polymer.pm	
	Result/PriorTherapy.pm	
	Result/Sample.pm	
	Result/Snpindel.pm	
	Result/SystemUser.pm	
	Result/SystemUserRole.pm	
	Result/TodoTask.pm	
	Result/TreatmentAssignment.pm	
	Result/TreatmentReport.pm	
	Result/Variant.pm	
/opt/applications/opengenemed/webroot/lib/OpenGeneMed	Controllers	classes for the different roles
	Plugins/Default.pm	support functions used by several other classes
	VCF	classes to support VCF upload and annotation
	VCF/AnnotateVCF.pm	main class for annotation
	VCF/File.pm	support file read class
	Nav.pm	tab and navigation functions
	Settings.pm	load configuration settings file

/opt/applications/opengenemed/webroot/lib/OpenGeneMed/Controllers	Admin	methods used by the Administrative Team
	Clinicteam	methods used by the Clinic Team
	Customize	methods used by the Study Coordinator to tailor the variant calling procedure and clinical information to the specific study needs.
	Labs	methods used by the Lab Team
	Stats	methods used by the Research and Biostatisticans Team
	Treatmentteam	methods used by the Treatment Team
	__Base__.pm	
	Admin.pm	gateway for admin tasks
	Clinicteam.pm	gateway for clinical team's tasks
	Customize.pm	gateway for study coordinator's tasks
	Dashboard.pm	methods to display information in the dashboard tab
	Labs.pm	gateway for lab team's tasks
	Profile.pm	methods to manage user profile
	Stats.pm	gateway for research statisticians' tasks
	Todo.pm	methods to manage notification requests across groups
	Treatmentteam.pm	gateway for treatment review team's tasks
/opt/applications/opengenemed/webroot/lib/OpenGeneMed/Controllers/Admin	Users.pm	manage user accounts
/opt/applications/opengenemed/webroot/lib/OpenGeneMed/Controllers/Clinicteam	Assignedt.pm	visualize assigned treatments
	Eligibility.pm	manage eligibility questions and answers
	Patients.pm	manage patient clinical information
	Priort.pm	manage prior therapies
	Samples.pm	visualize patient samples
	Treatment.pm	manage treatment assignment
/opt/applications/opengenemed/webroot/lib/OpenGeneMed/Controllers/Customize	ActionableRules.pm	manage custom actionable rules
	Amoi.pm	manage custom actionable mutation of interest table
	Eligibility.pm	manage custom eligibility questions
	GeneDrugs.pm	manage custom gene/drugs table
/opt/applications/opengenemed/webroot/lib/OpenGeneMed/Controllers/Labs	Patients.pm	visualize patient information
	Samples.pm	manage upload and review of samples
	Trialstatslab.pm	visualize statistics about the study (enrollment, variants distribution, ...)

/opt/applications/opengenemed/webroot/lib/OpenGeneMed/Controllers/Stats	Trialstats.pm	visualize statistics about the study (enrollment, variants distribution, ...)
/opt/applications/opengenemed/webroot/lib/OpenGeneMed/Controllers/Treatmentteam	Patients.pm	visualize patient information and assign treatments
/opt/applications/opengenemed/webroot/templates	admin	html templates used by the Administrative Team
	clinicteam	html templates used by the Clinic Team
	customize	html templates used by the Study Coordinator to tailor the variant calling procedure and clinical information to the specific study needs.
	labs	html templates used by the Lab Team
	layouts	utility template layout
	profile/index.html.ep	user profile page
	stats	html templates used by the Research and Biostatisticans Team
	todo	html templates used by the Treatment Team
	treatmentteam	html templates used by the Treatment Team
	not_found.html.ep	page for not found paths
	exception.html.ep	page for errors in code/display
	index.html.ep	template for dashboard page
	login.html.ep	html template for the login page
/opt/applications/opengenemed/webroot/templates/admin	users/create.html.ep	
	users/delete.html.ep	
	users/edit.html.ep	
	users/index.html.ep	
/opt/applications/opengenemed/webroot/templates/clinicteam	assignedt/index.html.ep	
	eligibility/edit.html.ep	
	eligibility/index.html.ep	
	eligibility/view.html.ep	
	patients/create.html.ep	
	patients/delink.html.ep	
	patients/delink_finalize.html.ep	
	patients/edit.html.ep	
	patients/index.html.ep	
	priort/create.html.ep	
	priort/edit.html.ep	
	priort/index.html.ep	
	priort/view.html.ep	
	samples/create.html.ep	
	samples/edit.html.ep	
	samples/index.html.ep	
	treatment/create.html.ep	
	treatment/edit.html.ep	

	treatment/index.html.ep	
	treatment/view.html.ep	
/opt/applications/opengenemed/webroot/ templates/customize	actionable_rules/create.html.ep	
	actionable_rules/delete.html.ep	
	actionable_rules/edit.html.ep	
	actionable_rules/index.html.ep	
	actionable_rules/ readablequery.html.ep	
	amoi/create.html.ep	
	amoi/delete.html.ep	
	amoi/edit.html.ep	
	amoi/index.html.ep	
	amoi/upload.html.ep	
	eligibility/create.html.ep	
	eligibility/delete.html.ep	
	eligibility/edit.html.ep	
	eligibility/index.html.ep	
	gene_drugs/create.html.ep	
	gene_drugs/delete.html.ep	
	gene_drugs/edit.html.ep	
	gene_drugs/index.html.ep	
/opt/applications/opengenemed/webroot/ templates/labs	index.html.ep	
	patients/delink.html.ep	
	patients/index.html.ep	
	patients/view.html.ep	
	samples/index.html.ep	
	samples/review.html.ep	
	samples/ reviewcompleted.html.ep	
	samples/view.html.ep	
	trialstatslab/allvariants.html.ep	
	trialstatslab/index.html.ep	
	trialstatslab/ instudyvariants.html.ep	
/opt/applications/opengenemed/webroot/ templates/layouts	content_box.html.ep	utility template used within default.html.ep
	default.html.ep	template with general page layout, header, tabs, libraries, styles, etc.
	naventry.html.ep	utility templated used to build the layout of each page
/opt/applications/opengenemed/webroot/ templates/stats	trialtstats/allvariants.html.ep	
	trialtstats/instudyvariants.html.ep	
	trialtstats/index.html.ep	
/opt/applications/opengenemed/webroot/ templates/todo	index.html.ep	
/opt/applications/opengenemed/webroot/ templates/treatmentteam	patients/assignt.html.ep	
	patients/clinicalinfo.html.ep	

	patients/currentt.html.ep	
	patients/eligibility.html.ep	
	patients/index.html.ep	
	patients/patientvariants.html.ep	
	patients/priort.html.ep	
	patients/variants.html.ep	
/opt/applications/opengenemed/webroot/ public	css	
	images	
	js	
	upload	

Key behaviors of the system are controlled through the settings stored in the `settings.ini` file (located in the config/local directory). Changes in this file require a restart of the opengenemed service to be effective. The file controls:

- debug options as the log directory and how snpEff is executed
- directories where different files are stored and uploaded
- information to login into mysql database
- information about the notification emails sent for different scenarios. They can be disabled (for debug purposes), their 'from' field can be customized and all emails can be redirected to one or more addresses (instead of the intended recipients) for debug purposes.
- the name of the application (displayed in the header of every page)
- activation/inactivation of the delink functionality

3.9 Customizing the code

A powerful feature of the OpenGeneMed system is the modular design of the source code and the ability of easily customize it to specific needs. The following sections illustrate this ability by walking the reader, step by step, into different customization of the system: changes are presented to the initial code and instructions on which files those changes affect are listed.

Customization can be classified into the following categories (ranked from the simplest to the most complex):

1. minor change in the back end: requires changes in just few methods, doesn't require changes in web interface;
2. major change in the back end: requires changes in several files and/or in the database structure, doesn't require changes in web interface;
3. minor change in the front end: requires changes in the web interface, doesn't require changes in back end;
4. major change in front and back end: require changes in both web interface and several files and/or database.

An example for each customization is presented in the next sections. Only the main details are outlined, but it should be sufficient for any skilled developer to use these guidelines to customize the system for different need.

3.9.1 Add validation rules for Patient ID

The Patient ID format is checked before the new patient is entered in the system by a member of the Clinic team. In the default implementation of OpenGeneMed the requirements for this ID are: should only contains letters, numbers or underscore and should be between 2 and 30 characters long. Different studies usually require a strict format for this ID since it may contain reference to specific components of the study and it's really important for the system to be able to formally validate this entry and minimize human error.

Adding/changing the validation rules for the patient id is very straightforward in the OpenGeneMed code. Below the extract of the code showing how the validation is currenly carried out and where it can be modified.

```
###
# Extract from Controllers/Clinicteam/Patients.pm
# Code called to add a new patient in the system
##
sub create {
 [...]
  if ( $self->req->method eq 'POST' ) {
    my $checkInput = $self-> _sanitizeAndCheckInput();
    [...]
    if ( $checkInput == 0 ) {
      my $user = $OpenGeneMed::DB->resultset('Patient')->new_result({
        pid => $self->stash->{pid},
        [...]
       })->insert;
      [...]
    }
  }
}

sub _sanitizeAndCheckInput() {
   [...]
  my $pid =   ( defined $self->param('pid') ) ? clean_XSS( $self->param('pid') ) : '';
   [...]
  if (length($pid) == 0 ) {
    $self->stash->{errorPID} = "Patient ID cannot be empty.";
    $status += 1;
  }
  else {
    if ( $pid !~ /^[A-Z0-9_]{2,30}$/i ) {
```

```
      $self->stash->{errorPID} = 'Patient ID must contain just letters, numbers...';
      $status += 1;
    }
    ###
    # Insert additional rules HERE
    ###
  }
   [...]
}
```

OpenGeneMed code extract 1: *Code extract for Patient ID validation. Only the relevant code is shown ('...' indicates omitted sections). Red and boldface methods shows the flow of execution. The current matching rule is also shown.*

3.9.2 Customize annotation steps (supporting different VCF versions)

A key component of OpenGeneMed is the capability of annotating VCF files and present the results of the annotation in a clear and concise way. In the default implementation of OpenGeneMed, only one specific version of VCF file is supported (version 4.0). It's easy to envision the need of different versions and formats for those files that are the results of existing bioinformatics pipelines.

The modular design of OpenGeneMed allow to easily change the parsing of those input files. Providing only a single different parsing function as shown below, allow the new input format to be fed seamlessly to the next steps of the annotation process.

```
###
# Extract from VCF/AnnotateVCF.pm (Code called when a VCF file is uploaded by a member of the lab team)
##
sub process {
  [...]
  my $validFile = $self->prepareVCF4SnpEff_version40($vcf);
  if ( $validFile == 1 ) {
    my $autosid = $self->sampleCreate();
    $self->runSnpEff( $self->{sampleID} );
    $self->processSnpEffNonsyn_version40();
    [...]
    $self->applyRules();
  }
  [...]
}

sub prepareVCF4SnpEff_version40 {
  my $vcf  = shift
  # check existance of ALT, REF column
  my $tmpFileName = "[...]{sampleID}.check";
  for (each line of $vcf) {
    #create new row in $tmpFileName with single ALT value
    #extract id, frequency, and other information for that specific entry and fill appropriate columns
  }
}

sub processSnpEffNonsyn_version40  {
  my $infile =  "[...]{sampleID}.snpEff.nonsyn";
  for ( each line of $infile ) {
    # using gsymbol, query the Gpd table to annotate pathway, drug and moitype
    # extract read depth, allele frequency, reason for no call, position,
    # using cosmic id, quert the Moi7 table to get information of additional cosmic id and counts
    # query the Polymer table to evaluate if the variant is in close proximity to an homopolymer region
    # query the Snpindel table to record information in the fields maf1, maf2, maf3
    # collect all this information and create an entry in the Variant table for this sample/patient/variant
}
```

OpenGeneMed code extract 2: *Code and pseudocode extract for variants annotation. Only the relevant code is shown ('...' indicates omitted sections). Red and boldface methods shows the flow of execution. Preprocessing and postprocessing of vcf*

intermediate files require detailed knowledge of the specific format and only one version is implemented in OpenGeneMed. The modular architecture of the code allows easy replacement of these functions with similar methods acting on a different format of VCF.

It's important to note that replacing the parsing function will allow only the new version to be annotated. If multiple versions are used, the code needs to be changed to account for those options both in the back end (the methods shown above) and in the front end (i.e., on the page where the Lab team member performs the upload, a drop-down menu should be presented to allow the selection of the appropriate format). Changes in the front end of OpenGeneMed are presented in the next two sections below.

3.9.3 Rearrange order of elements in the Clinical Information page

The look of all the web pages in OpenGeneMed can be customized to specific requirements. The rendering of the different pages is controlled by Mojolicious templates: even if this section will not go in the details of how those templates work, the general logic is pretty simple and the main steps to customize the clinical information page (rearranging the order of different elements) is outlined below.

Every action/method in the Patients controller (Patients.pm) is connected to the respective *Embedded Perl* (ep) template. The templates are located in the webroot/templates/clinicteam/patients folder, and their name represents the action they are rendering (e.g., ***edit.html.ep*** is the template we are going to alter for this tutorial, since this template controls the page where the clinical information of a patient is displayed to allow editing.

The file contains a mix of html/perl/javascript/css: in a nutshell the .ep file is a regular html file (with css/javascript) and added capabilities of accessing objects and variable using perl.

```
% layout 'default', 'title' => 'OpenGeneMed - Edit patient';
<script type="application/x-javascript">
[...]
</script>
[...]
<div style="text-align:left"> <b>Patient Clinical Information</b></div>
[...]

<form id="register_patient" class="form-horizontal" method="post" action="<%= stash 'url' %>">
  <table>
    <tr><td align="right"> Histologic Dx</td>
        <td align="center"> <input name="histologic_dx" type="text" value="<%=  $patient->dx %>" /></td>
    <tr><td align="right"> Histologic Dx Subtype</td> [...]
    <tr><td align="right"> Date of Diagnosis</td> [...]
    <tr><td align="right"> Age at Diagnosis</td> [...]
  [...]
  </table>
</form>
[...]
```

The document is structured as follows:

- it calls a 'default' template (located in the templates/layout directory) that contains the header for the default page. This template contains all the javascripts, stylesheets, etc that define the main look of the pages. This files calls several other 'sub-templates' (e.g., content_box) that all come together to build the final page displayed in OpenGeneMed. While the architecture may seem complicated, it's modularity allows, from one side, to seamlessly apply changes to basic common features (all contained in this default template) and, on the other side, focusing the render of a specific page (like the editing one) only to the elements relevant to this page (i.e., the clinical information of the patient)
- additional scripts can be added in the current template to render, for example, datatables and other objects specific to this page
- standard html elements that compose the elements of the page are listed, an example div is shown in the code

above

- the main part of the page is the form that contains the clinical information fields. Important to note are the reference to *'stash variables'* url and patient. These are variables set in the Mojolicious request to GET the 'edit' page and they are filled within the code in Patients.pm/edit. Since the action of posting the form is directed to the same url, all the values input by the user will be directed again to the Patients.pm/edit for processing
- the *patient* variable is a special object generated by a query on the database of patients. Details about how the database is handled in OpenGeneMed are presented in the next section

To customize the look of this page, just edit the html in this file and save: for example moving the "tr" tags in a different order would accomplish the task of reordering the elements in the clinical information page front end.

☞ Note: in our implementation, we take advantage of several standard libraries for html elements, like datatables and d3, each coming with a set of customization and initial setup. Refer to the official manuals/tutorials of the different libraries for more details on their specific usage.

3.9.4 Add information in the Patient Registration form

Customizing front and back end to add new features is quite straightforward even if it implies changes in several parts of the code.

In this section we present an example that requires changes in the web interface, the perl objects and the database. The example scenario is the following: we want to add more information in the Patient Registration form, to record race/ethnicity data for each patient.

First we need to understand what exactly we want to add: for this example we will add a drop down option to record the hispanic/not hispanic ethnicity and a list of checkboxes to record racial categories.

Below the actions needed to implement the change are listed in chronological order. Note that once the changes are made, the mojolicious morbo server should automatically restart and the changes should be accessible to the end user without further actions.

1. **Add columns in database**. Two new columns need to be added to the 'patient' table to store the new information. In mysql issue the following commands to perform the change.
   ```
   alter table patient add column ethnicity varchar(50);
   alter table patient add column race varchar(200);
   ```
   Sice the 'race' field will contain a comma separated list of values, we want to allow this field to be larger enough to host this information.

2. **Update Perl DBIx classes**. Since we are using DBIx classes to access the database, we need to add the new columns to the 'webroot/lib/Schema/Result/Patient.pm' file as follows:

```
[...]
=head1 ACCESSORS
[...]

=head2 ethnicity
 data_type:  'varchar'
  is_nullable:  1
  size:  50

=head2 race
  data_type:  'varchar'
  is_nullable:  1
  size:  200
=cut
```

```
__PACKAGE__->add_columns(
  "pid",
  [...]
  "randomid",
  { data_type => "varchar", is_nullable => 0, size => 20 },
  "ethnicity",
  { data_type => "varchar", is_nullable => 1, size => 50 },
  "race",
   { data_type => "varchar", is_nullable => 1, size => 200 }
);

=head1 PRIMARY KEY
[...]
```

3. **Change the web interface to include the new fields**. As mentioned in the previous section, the rendering of the different pages is controlled by Mojolicious templates. The templates for the patient clinical information are located in the webroot/templates/clinicteam/patients folder, and their name represents the action they are rendering (e.g., ***create.html.ep*** is the template we are going to alter in this step, since this template controls the page where the clinical information of a new patient is displayed. Note that similar actions need to be reproduced in the edit.html.ep if you want the 'Edit' page to display the new fields).

   The file contains a mix of html/perl/javascript/css: in a nutshell the .ep file is a regular html file (with css/javascript) and added capabilities of accessing objects and variable using perl.

```
% layout 'default', 'title' => 'OpenGeneMed - Register patient';
<script type="application/x-javascript">
[...]
</script>
[...]
<div style="text-align:left"> <b>Patient Clinical Information</b></div>
[...]

<form id="register_patient" class="form-horizontal" method="post" action="<%= stash 'url' %>">
  <table>
    <tr><td align="right"> Histologic Dx</td> [...]
    <tr><td align="right"> Histologic Dx Subtype</td> [...]
    <tr><td align="right"> Date of Diagnosis</td> [...]
    <tr><td align="right"> Age at Diagnosis</td> [...]
    <tr><td align="right"> Ethnicity</td>
        <td align="center"> <select name="ethnicity" type="select" >
        <%=$self->_getEthnicity_optionTags();=%>
        </select>
        </td><td style="color:red"><%= stash 'errorEthnicity' %></td></tr>
    <tr><td align="right"> Race</td>
        <tr><td align="right"> Racial Categories <br/>(select one or more, if applicable)</td>
        <td align="left">
        <%=$self->_getRace_checkboxes();=%>
        </td><td style="color:red"><%= stash 'errorRace' %></td></tr>

  [...]
  </table>
</form>
[...]
```

   Note that the additional drop down and check boxes make use of functions in the Patients.pm file whose changes are listed below.

4. **Change the patient registration code to record the new fields**. The last step require modifications to the Patients.pm file, stored in the Clinicteam directory.

```
###
# Extract from Clinicteam/Patients.pm
##
sub create {
  [...]
  elsif ( $self->req->method eq 'POST' ) {
    my $checkInput = $self->_sanitizeAndCheckInput();

    my $ethnicity= (defined $self->param('ethnicity')) ?  clean_XSS($self->param('ethnicity')) :  '';
    my $params = $self->req->params->to_hash();
    $self->paramToArrayRef($params, 'race');
    my race_list = $params->'race';
    my $race = join( ',', race_list);
    $race = clean_XSS($race);
    [...]

    if ( $checkInput == 0 ) {
        my $user = $OpenGeneMed::DB->resultset('Patient')->new_result({
          date_submit => \'NOW()',
          pid => $self->stash->{pid},
          dx => $self->stash->{histologic_dx},
          dx_subtype => $self->stash->{histologic_dx_subtype},
          dx_date => $self->stash->{date_diagnosis},
          age_dx => $self->stash->{age_diagnosis},
          sex =>  $self->stash->{gender},
          comments =>  $self->stash->{comments},
          race => $race,
          ethnicity => $ethnicity
        })->insert;
  [...]
}

sub _getEthnicity_optionTags {
      my $self = shift;
      my $selected = shift || '';

      my $output = '';
      if ( $selected eq '' ) { $output .= "<option value=\"\" selected></option>";}
      else { $output .= "<option value=\"\"></option>"; }

      $output .= "<option value='Hispanic or Latino' ";
      if ( $selected eq 'Hispanic or Latino' ) { $output .= " selected "; }
      $output .= ">Hispanic or Latino</option>";

      $output .= "<option value='Not Hispanic or Latino' ";
      if ( $selected eq 'Not Hispanic or Latino' ) { $output .= " selected "; }
      $output .= ">Not Hispanic or Latino</option>";

      $output .= "<option value='Unknown' ";
      if ( $selected eq 'Unknown' ) { $output .= " selected "; }
      $output .= ">Unknown</option>";

      $output .= "<option value='Declined to answer' ";
      if ( $selected eq 'Declined to answer' ) { $output .= " selected "; }
      $output .= ">Declined to answer</option>";

      return Mojo::ByteStream->new($output);
  }

sub _getRace_checkboxes {
      my $self = shift;
      my $selected = shift || '';

      my $output = '';
```

```
    $output .= "<input style='vertical-align: top' type='checkbox'
                name='race' value='American Indian/Alaska Native' ";
    if ( $selected eq 'American Indian/Alaska Native' ) { $output .= " checked "; }
    $output .= ">  American Indian/Alaska Native <br>";

    $output .= "<input style='vertical-align: top' type='checkbox' name='race' value='Asian' ";
    if ( $selected eq 'Asian' ) { $output .= " checked "; }
    $output .= ">  Asian <br>";

    $output .= "<input style='vertical-align: top' type='checkbox'
                name='race' value='Native Hawaiian or Other Pacific Islander' ";
    if ( $selected eq 'Native Hawaiian or Other Pacific Islander' ) { $output .= " checked "; }
    $output .= ">  Native Hawaiian or <br>      Other Pacific Islander <br>";

    $output .= "<input style='vertical-align: top' type='checkbox'
                name='race' value='Black or African American' ";
    if ( $selected eq 'Black or African American' ) { $output .= " checked "; }
    $output .= ">  Black or African American <br>";

    $output .= "<input style='vertical-align: top' type='checkbox' name='race' value='White' ";
    if ( $selected eq 'White' ) { $output .= " checked "; }
    $output .= ">  White <br>";

    $output .= "<input style='vertical-align: top' type='checkbox' name='race' value='Unknown' ";
    if ( $selected eq 'Unknown' ) { $output .= " checked "; }
    $output .= ">  Unknown <br>";

    $output .= "<input style='vertical-align: top' type='checkbox'
                name='race' value='Declined to answer' ";
    if ( $selected eq 'Declined to answer' ) { $output .= " checked "; }
    $output .= ">  Declined to answer <br>";

    return Mojo::ByteStream->new($output);

}
```

The code makes use of two functions (*_getEthnicity_optionTags* and *_getRace_checkboxes*) to promote modularity and maintainability in the code design: the answers for these questions could be stored in additional tables in the database or external files, to allow easy changes/additions. For the purpose of this tutorial, since we wanted to focus on the general code modifications and not on these specific answers, we choose to hardcode all the different answers.

References

Bamford, S., Dawson, E., Forbes, S., Clements, J., Pettett, R., Dogan, A., Flanagan, A., Teague, J., Futreal, P. A., Stratton, M., et al. (2004). The cosmic (catalogue of somatic mutations in cancer) database and website. *British journal of cancer*, 91(2):355–358.

Doroshow, J. H. (2010). Selecting systemic cancer therapy one patient at a time: is there a role for molecular profiling of individual patients with advanced solid tumors? *Journal of Clinical Oncology*, 28(33):4869–4871.

Gargis, A. S., Kalman, L., Berry, M. W., Bick, D. P., Dimmock, D. P., Hambuch, T., Lu, F., Lyon, E., Voelkerding, K. V., Zehnbauer, B. A., et al. (2012). Assuring the quality of next-generation sequencing in clinical laboratory practice. *Nature biotechnology*, 30(11):1033–1036.

Kummar, S., Williams, P. M., Lih, C.-J., Polley, E. C., Chen, A. P., Rubinstein, L. V., Zhao, Y., Simon, R. M., Conley, B. A., and Doroshow, J. H. (2015). Application of molecular profiling in clinical trials for advanced metastatic cancers. *Journal of the National Cancer Institute*, 107(4):djv003.

Rothberg, J. M., Hinz, W., Rearick, T. M., Schultz, J., Mileski, W., Davey, M., Leamon, J. H., Johnson, K., Milgrew, M. J., Edwards, M., et al. (2011). An integrated semiconductor device enabling non-optical genome sequencing. *Nature*, 475(7356):348–352.

Simon, R. and Polley, E. (2013). Clinical trials for precision oncology using next-generation sequencing. *Personalized Medicine*, 10(5):485–495.

Simon, R. and Roychowdhury, S. (2013). Implementing personalized cancer genomics in clinical trials. *Nature reviews Drug discovery*, 12(5):358–369.

Tran, B., Brown, A. M., Bedard, P. L., Winquist, E., Goss, G. D., Hotte, S. J., Welch, S. A., Hirte, H. W., Zhang, T., Stein, L. D., et al. (2013). Feasibility of real time next generation sequencing of cancer genes linked to drug response: results from a clinical trial. *International Journal of Cancer*, 132(7):1547–1555.

Vogelstein, B., Papadopoulos, N., Velculescu, V. E., Zhou, S., Diaz, L. A., and Kinzler, K. W. (2013). Cancer genome landscapes. *science*, 339(6127):1546–1558.

Zhao, Y., Polley, E., Li, M.-C., Lih, C. J., Palmisano, A., Sims, D. J., Rubinstein, L. V., Conley, B. A., Chen, A. P., Williams, M. P., Kummar, S., Doroshow, J. H., and Simon, R. M. (2015). GeneMed: an informatics hub for the coordination of next-generation sequencing studies that support precision oncology clinical trials. *Cancer Informatics*, 14(Suppl 2).