Project Stack

* Framework : Spring 4
* ORM : Hibernate 4
* Other libraries : Log4j,Joda Time, Velocity Template Engine, Jasypt, Bouncy Castle,

Calendar API Client, Geo Lite, Jackson etc.

* Frontend : HTML, CSS, Bootstrap, JQuery, Mustache Js
* IDE : Eclipse Mars
* Language : JAVA 7
* Server : Tomcat 7
* Version control : GitHub
* Build Tool : Maven
* Database : MySQL
* Hosting : Godaddy VPS

Note: Please refer maven pom.xml to know the exact versions of libraries that we used in the project.

Things to understand

1. Basic knowledge on concepts of Spring Framework 4 like Core, Context, Beans, MVC, Mail and Security .We used purely java based configuration with annotations to configure the spring application instead of using xml based configuration. Integrated the Spring Security feature with Spring web MVC. Integrated Hibernate 4 with Spring Framework.

Reference: <https://spring.io/>

<http://docs.spring.io/spring/docs/4.2.6.RELEASE/spring-framework-reference/htmlsingle/>

1. Basic knowledge on concepts of Hibernate 4. We used criteria API for querying the database.

Reference: <http://hibernate.org/orm/documentation/4.3/>

1. Should know how to integrate and configure Log4j. Check \src\main\resources\log4j.properties file.
2. We used Jasypt, Bouncy Castle libraries for encryption and decryption. We integrated it into Hibernate and Spring context with the help of dependency injection.

Reference: <http://www.jasypt.org/>

Reference: <https://www.bouncycastle.org/>

Most of the properties in application.properties file are encrypted for security reasons .And they are decrypted using Jasypt configuration.

Reference: <http://www.jasypt.org/encrypting-configuration.html>

You can also decrypt or encrypt a string using jasypt command line tool which is handy.

Reference: <http://www.jasypt.org/cli.html>

1. We used Google Calendar Client API to create and update events on google calendar. We are establishing a server to server communication using a Personal Information Exchange (.p12) certificate file

Reference: <https://developers.google.com/api-client-library/java/>

1. We used Spring Framework’s email library for sending emails.

Reference: [http://docs.spring.io/spring/docs/4.2.6.RELEASE/spring-framework-reference/htmlsingle/#mail](http://docs.spring.io/spring/docs/4.2.6.RELEASE/spring-framework-reference/htmlsingle/%23mail)

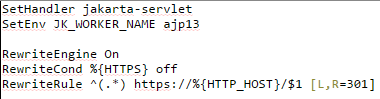
1. We used Velocity Template Engine to generating templates for sending transactional emails.

Reference: <http://velocity.apache.org/engine/devel/user-guide.html>

1. We used Joda Time library instead of using java.util.Date.
2. We used Jackson library to serialize the objects to JSON in Rest Controllers.
3. We used Geo Lite library to extract the user location based on IP address.

Reference: <http://dev.maxmind.com/geoip/legacy/geolite/>

1. We used @Async and @EnableAsync to send emails, create and delete google calendar events asynchronously.
2. We used @Scheduled annotation to schedule batch tasks like sending remainder emails.
3. We installed the SSL certificate on Godaddy server to make sure that all data passed between the web server and browsers remain private and integral. On GoDaddy hosting tomcat 7 is not installed as a standalone application. Instead, it runs on apache server module. So the SSL certificates are installed on apache instead of tomcat. So we can’t configure the tomcat 7 to run on secure port 8443 by editing the server.xml file. Instead, we should rewrite all the request URL to HTTPS using .htaccess file with the following code.



Refference: <https://www.godaddy.com/help/redirect-http-to-https-automatically-8828>

Since our application is deployed through Cpanel, the first two lines are necessary to allow the http request from the client to reach our application.

Reference: <https://documentation.cpanel.net/display/CKB/How+to+Deploy+Java+Applications>

1. We are renaming the file on upload to ensure the correct file extension. We are preventing direct access to uploaded files altogether. This way, any files uploaded to our web site are stored in a folder outside of the web root. We changed the file permissions to 0666 so it can't be executed.

For Implementation details refer the source code and Dentist\_Project\_Presentation.pdf file