ACWT – Tracking Accidents on Google Maps

Requirements and Objectives

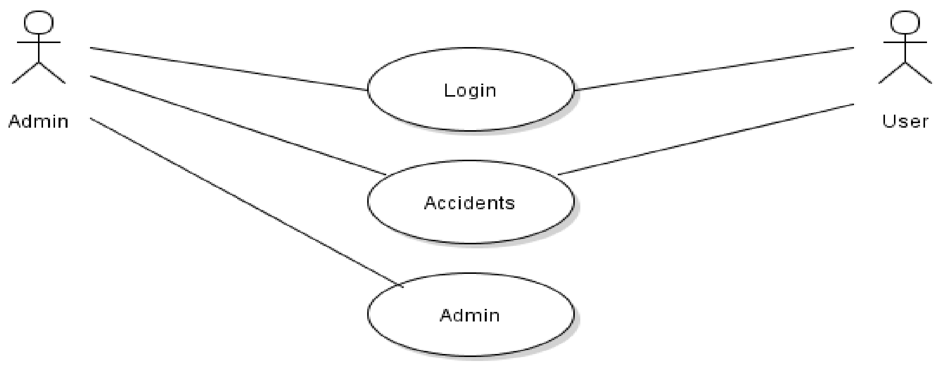
This system was designed to be a comprehensive road accident tracker within the city of London, and to allow users to submit their own accidents onto a secure website. The stakeholders in this case are people who commute within London or otherwise travel on the roads, as they have a vested interest in being able to track accidents and otherwise be able to view them in an easily accessible manner. This is similar to apps such as Waze, but by logging all accidents will show clusters rather than just incidents in the process of being resolved.

Objective one: The system will be able to log every single accident within the city of London in the past eight years. This data can be retrieved from TFL's website, but will be stored on the site in case it goes down.
(Entirely complete.)

Objective two: The system will be able to display these accidents as markers on a map of London. (This objective has been altered slightly, as unexpected constraints mean that all accidents cannot be displayed without unprecedented costs and effort).

Objective three: Users will be able to submit their own accidents to be logged on the site. (Unfulfilled due to time constraints, but the framework for doing so does exist).

System Architecture



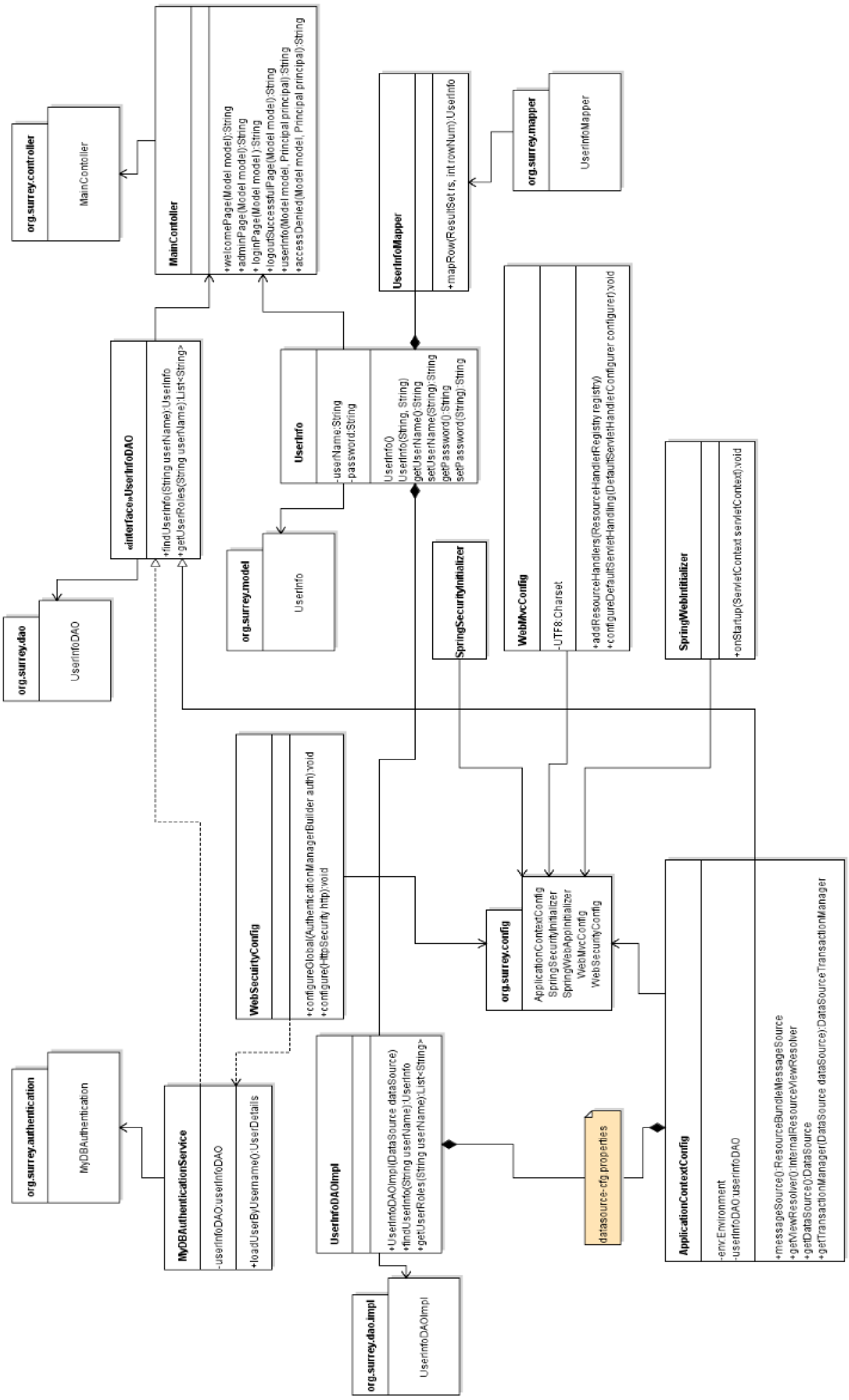
In this use case diagram, you can see the two actors for the website who are the admins and the users. Both the admins and users will be able to login to the system and will be able to access the accidents page where they will be able to see the accidents presented on the Google Maps. However only an admin will be able to access the admin page and if a user tries to access the admin page, then the system will show an error message informing the user they don't have the permission to access the admin page.

ii)

Class	Description
MyDBAuthenticationService.java	This class has been used to allow the user to login successfully to the website. It does this by matching the credentials entered in the website with the credentials saved in the database.

ApplicationContextConfig.java	This class has been used to load up the properties from the datastore file which has the database details.
SpringSecurityInitializer.java	This class extends AbstractSecurityWebApplicationInitializer.
SpringWebAppInitializer.java	This class implements WebApplicationInitializer and it registers the servlet at the start-up of the website.
WebMvcConfig.java	This class adds converters to the website for better handling of data.
WebSecurityConfig.java	This class adds the configuration for the webpages and the database.
MainController.java	This class connects the java classes to the JSP pages and gives content for the 403 page.
<Interface>UserInfoDAO.java	This interface gets the user info and gets the user roles as a list which can be used by other classes to authenticate the user and its role: user/admin.
UserInfoDAOImpl.java	This class contains the commands required for the MySQL database when data needs to be fetched from the data. This data will be used by other classes.
UserInfoMapper.java	This class contains method regard the MySQL database it returns the username and the password.
UserInfo.java	This class has the method which allows to get the username, password and also to set the username and password.

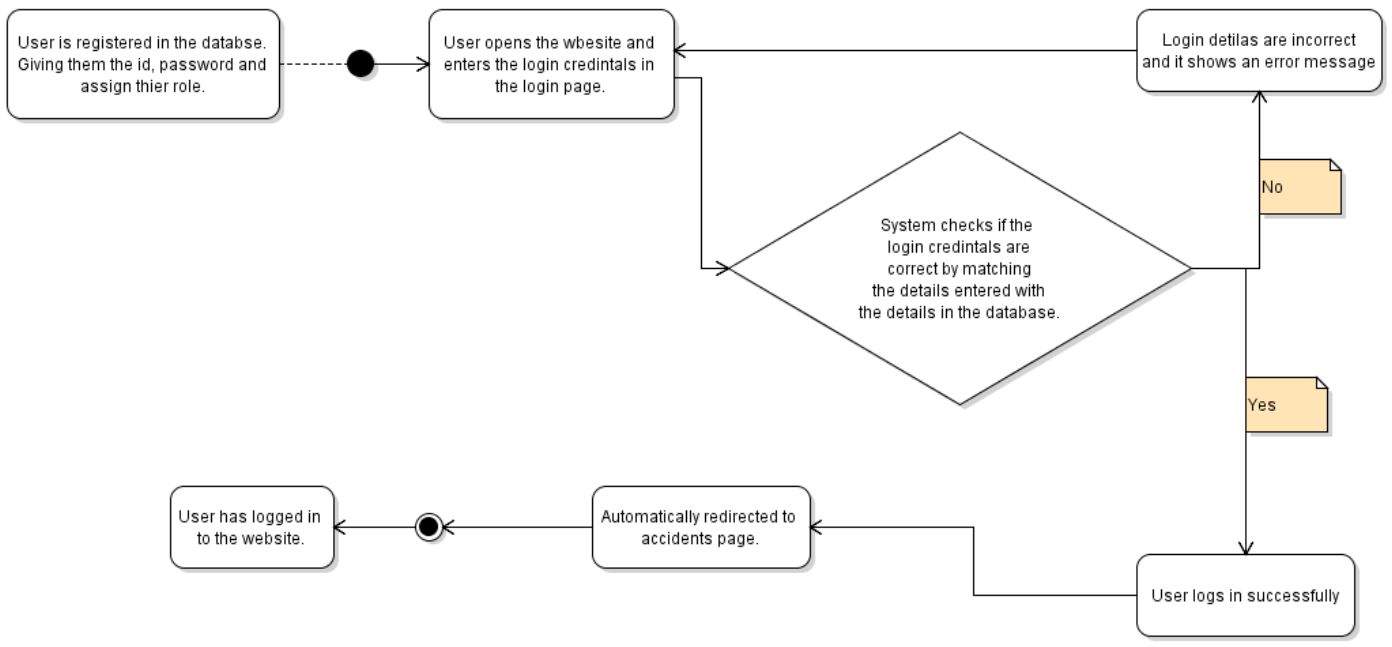
The package diagram below shows the different classes used to make the website work. It shows the fields and the methods used by each class. It also shows how the classes are related to each other.



org.surrey.controller
MainContoller
MainContoller
+welcomePage(Model model):String
+adminPage(Model model):String
+ loginPage(Model model):String
+logoutSuccessfulPage(Model model):String
+userInfo(Model model, Principal principal):String
+accessDenied(Model model, Principal principal):String
UserInfoMapper
+mapRow(ResultSet rs, int rowNum):UserInfo
org.surrey.mapper
UserInfoMapper
«interface»UserInfoDAO
+findUserInfo(String userName):UserInfo
+getUserRoles(String userName):List<String>
UserInfo
-userName:String
-password:String
UserInfo()
UserInfo(String, String)
getUserName():String
setUserName(String):String
getPassword():String
setPassword(String):String
org.surrey.model
UserInfo
org.surrey.dao
UserInfoDAO
SpringSecurityInitializer
WebMvcConfig
-UTF8:Charset
+addResourceHandlers(ResourceHandlerRegistry registry)
+configureDefaultServletHandling(DefaultServletHandlerConfigurer configurer):void
SpringWebInitializer
+onStartup(ServletContext servletContext):void
WebSecuirtyConfig
+configureGlobal(AuthenticationManagerBuilder auth):void
+configure(HttpSecurity http):void
org.surrey.config
ApplicationContextConfig
SpringSecurityInitializer
SpringWebAppInitializer
WebMvcConfig
WebSecurityConfig
org.surrey.authentication
MyDBAuthentication
MyDBAuthenticationService
-userInfoDAO:userInfoDAO
+loadUserByUsername():UserDetails
UserInfoDAOImpl
+UserInfoDAOImpl(DataSource dataSource)
+findUserInfo(String userName):UserInfo
+getUserRoles(String userName):List<String>
datasource-cfg.properties
ApplicationContextConfig
-env:Environment
-userInfoDAO:userInfoDAO
+messageSource():ResourceBundleMessageSource
+getViewResolver():InternalResourceViewResolver
+getDataSource():DataSource
+getTransactionManager(DataSource dataSource):DataSourceTransactionManager
org.surrey.dao.impl
UserInfoDAOImpl

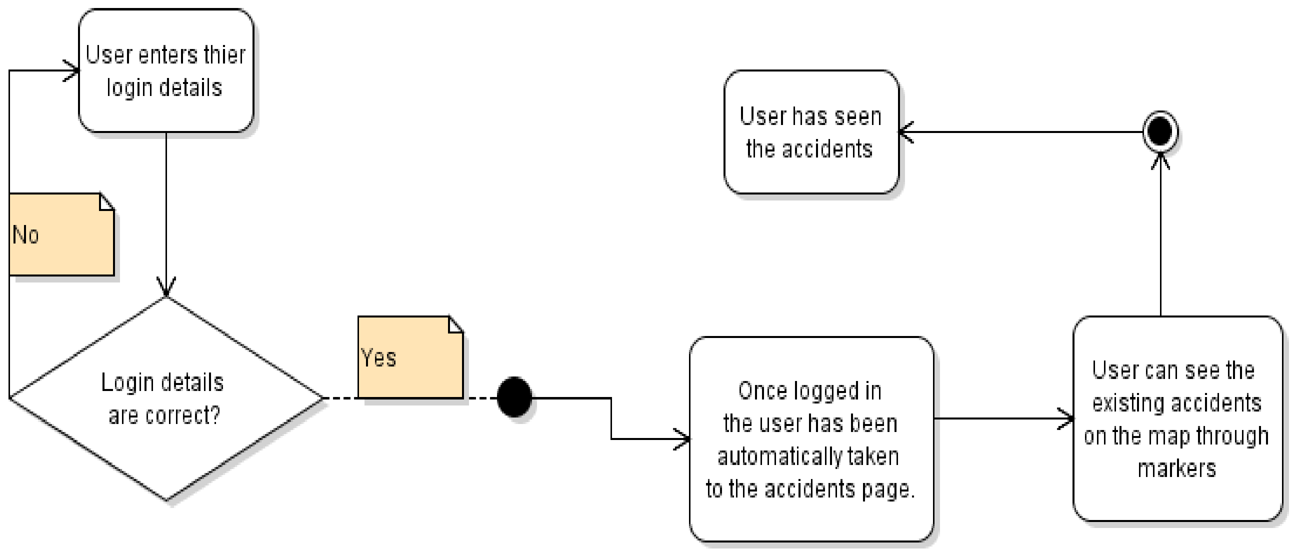
ii) 3

Login Process


User is registered in the databse. Giving them the id, password and assign thier role.
User opens the wbesite and enters the login credintals in the login page.
Login detilas are incorrect and it shows an error message
System checks if the login credintals are correct by matching the details entered with the details in the database.
No
Yes
User logs in successfully
Automatically redirected to accidents page.
User has logged in to the website.

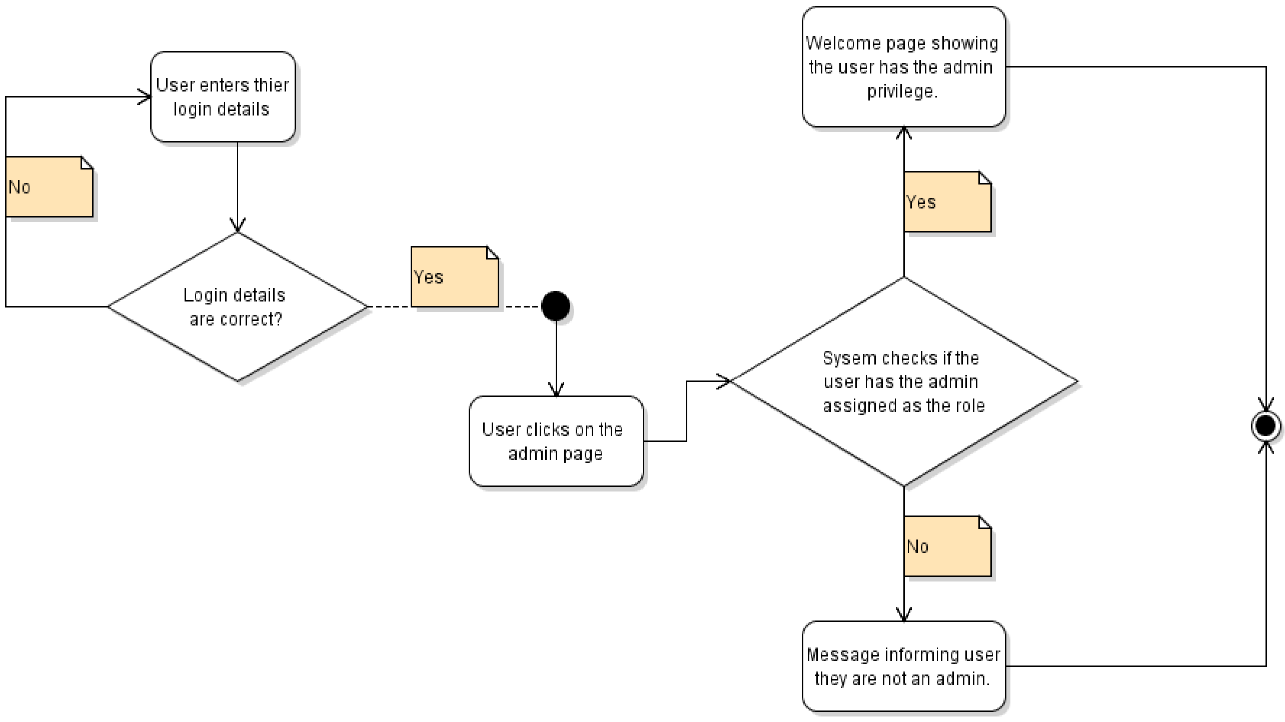
This diagram above shows the activity diagram for the login process. It shows step by step how the process takes place. At first a user gets registered to the database, so they are given a username, password and they accounts gets a role assigned either admin or user. Once the user has the login credentials he/she will enter those details on the website login page. The system checks if the user exists in the database. If it does matches, then the user logs in successfully and taken to the accidents page where the user can see the existing accident in the area. If the login details don't match, then it gives an error to the user saying login details are incorrect and takes it back to the login page.

Accidents Page


User enters thier login details
No
Login details are correct?
Yes
Once logged in the user has been automatically taken to the accidents page.
User can see the existing accidents on the map through markers
User has seen the accidents

The diagram above shows the procedure of a user accessing the accidents page. It starts off with the user logging on first as if they are not logged in then they will be redirected to the login page when they click on the accidents page. So, the user enters the details and if they match then they are taken to the accidents page where they can see existing accident on google maps for the area. If the login details don't match, then it will take the user to the login page and inform the user that the login details don't match so try again.

<u>Admin Page</u>


User enters thier login details
No
Login details are correct?
Yes
User clicks on the admin page
Sysem checks if the user has the admin assigned as the role
Yes
Welcome page showing the user has the admin privilege.
No
Message informing user they are not an admin.

The diagram above shows the procedure taking place accessing the admin page. It starts off with the user logging on first as if they are not logged in then they will redirect to the login page when they click on the admin page. So, the user enters the details and if they match then they are taken to the admin page and then the system checks if the user logged in has the permission to access the admin page by checking the user role of the account. If the user has admin role then it will great with a message and will allow the user to access the admin page but if the user does not have the admin role it will be inform them they don't have the permission as their account is not admin. If the login details don't match, then it will take the user to the login page and inform the user that the login details don't match so try again.

Technologies and Patterns used

I have used the MVC patterns for our website. In this pattern the application is split into three concerns: Model, Views and Controller. Model represents an object carrying the data, view will represent the visualisation of the data in the model class and controller acts on both the model and the view. It will update the view pages whenever the data changes or updates in the model object as it carries the model object and view, but it keeps them separate.

UserInfo.java-Model, JSP Pages-View and MainContoller.java-Controller

UserInfo class acts as the model for this website as it contains the data in terms of the username and password. JSP Pages acts as the view for this website as it gets the information from the model class to the user. Maincontoller.java is the controller class as it acts in the middle between the model and view class.

Basic Security

To include basic security to the login, there is a login system which means the user needs to be authenticated before they can use the website as to access the accidents page in the website it will require an account to access it. Hence not anyone can just get the access to the accident page, only those who are able to login successfully.

Clear User Interface

I believe that the interface is very clear and easy to use as there is a menu navigator at the top of every webpage which allows the user to navigate the website easily and swiftly. The system does offer its services through web interface and the presentation logic is very well separated from the system's services.

Object Relational Mapping

JDBC driver has been used to get the data from the MySQL database and represent into the form that it could be read by the java system which can be used to access the database and match the login details in this scenario. This can be seen in the ApplciationContextConfig.java file which show the setting up of the data source to connect to the database.

Use of Ajax

Ajax has been used to get the accidents data which is dynamic on the Google Maps. It loads up the accidents as markers on the maps by getting the data from the server and showing it as markers. It will automatically change every time there there's an update as it pulls the data dynamically.

Spring Framework- MVC

Annotations has been used to setup the MVC of the application instead of the XML configuration setup files. This can be seen in the config package of the project. In this project it contains ApplicationContextConfig.java, SpringSecurityInitializer.java, SpringWebAppInitializer.java, WebMvcConfig.java and WebSecurityConfig.java classes. These classes have been used to setup the MVC configurations of this web app.

Server-side password-MySQL

To increase security there has been a username and a password setup by MySQL which needs to verify for authenticity reasons. The username and the password for the MySQL server has been mentioned in the system so it could access the database.

Deployment and user manual

Deployment Manual

Prerequisites

1. MySQL Server

2. Eclipse IDE for Java EE Developers
3. Tomcat Server Version 9 core zip file
4. Coursework zip file

To deploy an application by using a setup project

- Open MySQL server and create a database called mydatabase and create two tables: USERS and USER_ROLES.
- Use these MySQL commands to set up the tables in the database:

	ands
	create table USERS (USERNAME VARCHAR(36) not null, PASSWORD VARCHAR(36) not null, ENABLED smallint not null) ; alter table USERS add constraint USER_PK primary key (USERNAME);
ROLES	create table USER_ROLES (ROLE_ID VARCHAR(50) not null, USERNAME VARCHAR(36) not null, USER_ROLE VARCHAR(30) not null) ; alter table USER_ROLES add constraint USER_ROLE_PK primary key (ROLE_ID); alter table USER_ROLES nstraint USER_ROLE_UK unique (USERNAME, USER_ROLE);

- Populate the tables

	ands

	insert into users (USERNAME, PASSWORD, ENABLED)
	
	values ('dbuser1', '12345', 1);
	
	insert into users (USERNAME, PASSWORD, ENABLED)
	
	values ('dbadmin1', '12345', 1);

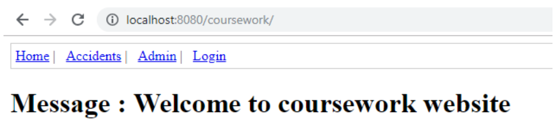
_ROLES	insert into User_Roles (ROLE_ID, USERNAME, USER_ROLE)
	
	values ('1', 'dbuser1', 'USER');
	
	insert into User_Roles (ROLE_ID, USERNAME, USER_ROLE)
	
	values ('2', 'dbadmin1', 'ADMIN');
	
	insert into User_Roles (ROLE_ID, USERNAME, USER_ROLE)
	
	('3', 'dbadmin1', 'USER');

- Open Eclipse, click on file and then select the option of open projects from file system and navigate to the coursework zip file and tick only the option to import as an eclipse project and click on finish.
- Expand the project and find the datasource-cfg.properties file and open it. Now change the configuration as to your MySQL configurations.
- To run the website, the user needs to install the server so click on the server's tab at the bottom and select on the option to add a server. If the user is not able to see the servers tab, then click on windows from the top menu and select show view and choose the serves option.
- Once the option pops up in a window choose the Tomcat v9.0 Server from the apache dropdown and click next. Now direct to the core zip file and select the JRE and then click finish. Now you should have the server installed.
- Right click on the project select run as and then select Run on Server and wait for the Tomcat to start and then it will show the website.
- If the user wants, then they can open it on a web browser also by going to this URL: (http://localhost:8080/coursework/).

User Manual

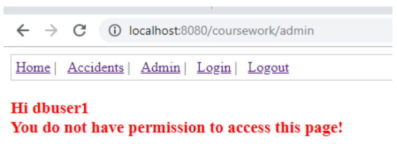
Once the site has been hosted and accessed, operation should be mostly simple. The first page you will be presented with is the welcome page, from which the login and admin pages can be accessed.

All pages require a login, however, and as such accessing the other pages from the welcome site will simply redirect you to an error page. As such, your first action should be to log in.


localhost:8080/coursework/

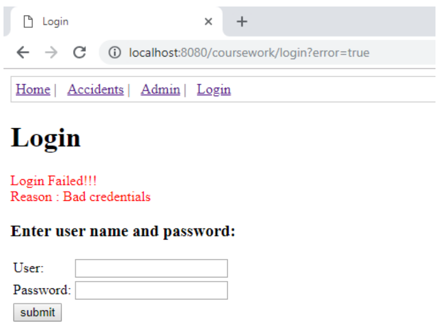
Home | Accidents | Admin | Login

Message : Welcome to coursework website


localhost:8080/coursework/admin

Home | Accidents | Admin | Login | Logout

Hi dbuser1
You do not have permission to access this page!


Login

localhost:8080/coursework/login?error=true

Home | Accidents | Admin | Login

Login

Login Failed!!!
Reason : Bad credentials

Enter user name and password:

User:
Password:
submit

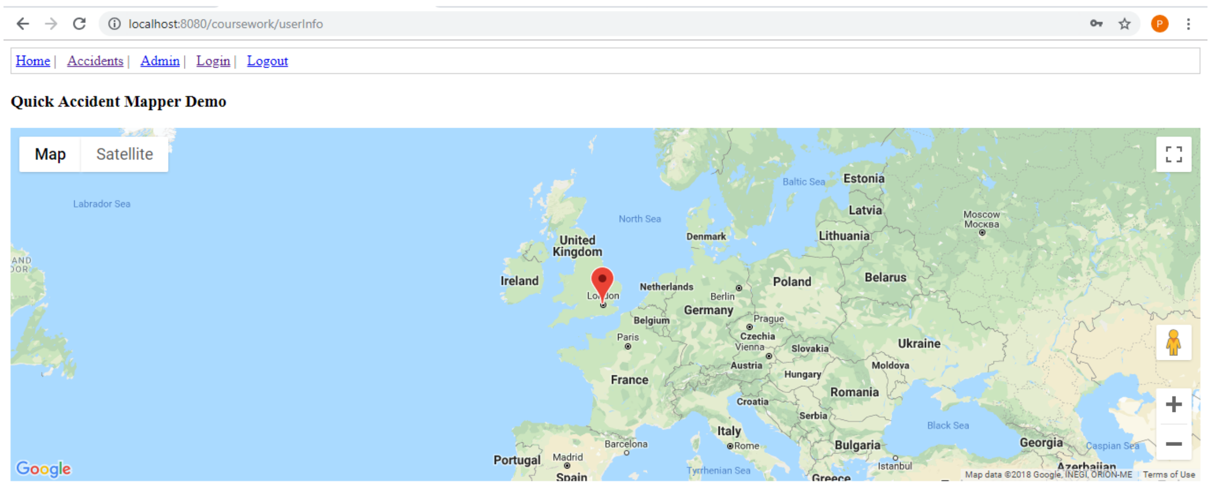
There are several logins that can be used:

Username	Password
dbuser1	12345
User1	12345

All user names and passwords are case sensitive. The first user has admin capabilities, while the second has limited permissions and cannot alter the database. The “admin” page is currently blank, but can only be accessed by users with administrative permissions.

For troubleshooting purposes, the database itself can also be accessed directly with the password root/root. It is possible to change the username and password of the admin user on the site using this and should be used if the site is compromised. Otherwise I would advise against it.

The accident page is displayed below. Simply zoom in to your preferred area of London, or out if you want to see all the marks clustered together on one point.


localhost:8080/coursework/userInfo

Home | Accidents | Admin | Login | Logout

Quick Accident Mapper Demo

Group roles and structure

Due to the small size of our group the structure was atypical – I (Lukas) took charge early on when it came to designation of work and the base design of the project, and afterwards we met weekly when possible to compare work done and test it together.

The GIT repository was stored at https://github.com/Lukasrygh23/AdvWebTech and used as a method of communicating files between us and emergency source control. Technical difficulties on Madan's half preventing him from directly pushing to the archive, so all files were provided to me and I committed them myself. This did lead to some delays however, and means all files are incorrectly attributed to me. We did both download from the archive to ensure proper version control, however.

Said archive can be used as a backup if there is some issue with the upload and will contain the most recent version.

Group Contributions

Lukas Rygh’s contribution

I contributed the initial design and suggestion for the project after early meetings and discussions, and focused primarily on the web implementation and overall design – alongside some work on database management. I was also in charge of managing source control and ensuring the google services cooperated – my account is being billed for all costs relating to the coursework, to use one example.

My biggest difficulty with the program was in attempting to convert the data I had retrieved into a form readable by google maps, and to attempt to cleanly translate this into database form.

Madan Pavadeep’s contribution

I took the responsibility of creating the website using java, spring and maven functionalities eventually and Lukas was responsible for the Google Maps integration with the website. I was also responsible for the database for the login system to work for which I used MySQL. I was helped by Lukas for this part.

My overall contribution for this project is the making of the website which uses maven dependencies and linking the Goggle Maps integration with the website and creating a database for the login system to work.

My contribution towards the report: I did the use case diagram including an explanation paragraph for the project which shows the usage process of the diagram. Moreover, I also provided the project class diagram including a paragraph explaining the java classes and usage. Furthermore, I also did the activity diagrams for the most three important functionalities of the website which was the login process, accidents page and admin role. I was also responsible for explaining the activity diagrams. I also provided explanations and description on the choice of technologies and patterns used in the implementation. At last I did the deployment manual of the system.

GROUP WORK EFFORT

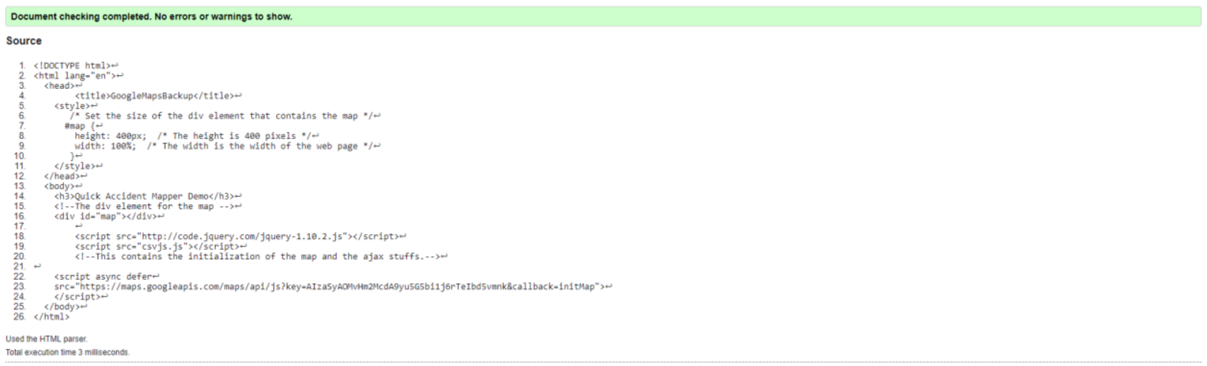
I, Lukas Rygh, believe work effort was equal for both members.

I Pavandeep Madan, believe work effort was equal for both members.

Appendix

HTML Verification

For userInfoPage.jsp:



Document checking completed. No errors or warnings to show.

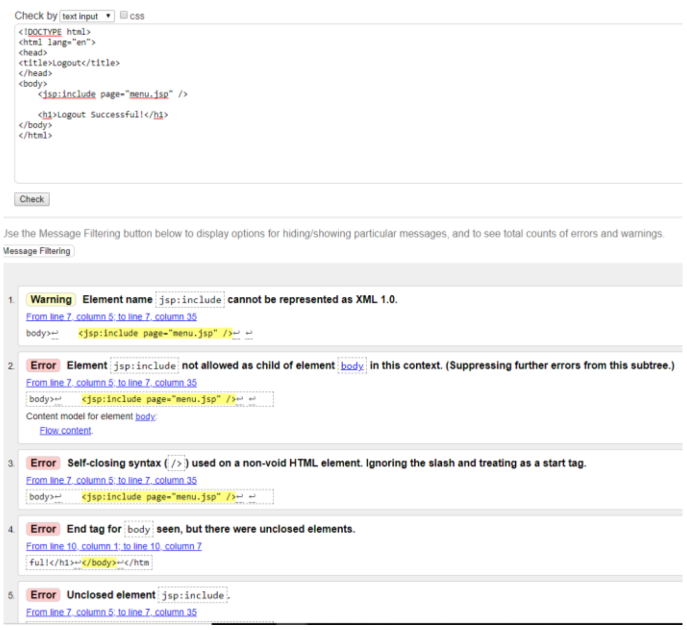
Source

```
<!DOCTYPE html>↩
<html lang="en">↩
  <head>↩
        <title>GoogleMapsBackup</title>↩
    <style>↩
       /* Set the size of the div element that contains the map */↩
      #map {↩
        height: 400px;  /* The height is 400 pixels */↩
        width: 100%;  /* The width is the width of the web page */↩
       }↩
    </style>↩
  </head>↩
  <body>↩
    <h3>Quick Accident Mapper Demo</h3>↩
    <!--The div element for the map -->↩
    <div id="map"></div>↩
        ↩
        <script src="http://code.jquery.com/jquery-1.10.2.js"></script>↩
        <script src="csvjs.js"></script>↩
        <!--This contains the initialization of the map and the ajax stuffs.-->↩
↩
    <script async defer↩
    src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAOMvHm2McdA9yu5G5bi1j6rTeIbd5vmnk&callback=initMap">↩
    </script>↩
  </body>↩
</html>
```

Used the HTML parser.
Total execution time 3 milliseconds.

All other pages misread /jsp and as such broke the validator. One example below.

logoutSuccessfulPage.jsp



Check by text input ▾ ☐ css

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Logout</title>
</head>
<body>
    <jsp:include page="menu.jsp" />

    <h1>Logout Successful!</h1>
</body>
</html>
```

Check

Use the Message Filtering button below to display options for hiding/showing particular messages, and to see total counts of errors and warnings.

Message Filtering

1. **Warning** **Element name** `jsp:include` **cannot be represented as XML 1.0.**
   From line 7, column 5; to line 7, column 35
   `body>↩     <jsp:include page="menu.jsp" />↩ ↩`
2. **Error** **Element** `jsp:include` **not allowed as child of element** `body` **in this context. (Suppressing further errors from this subtree.)**
   From line 7, column 5; to line 7, column 35
   `body>↩     <jsp:include page="menu.jsp" />↩ ↩`
   Content model for element body:
   Flow content.
3. **Error** **Self-closing syntax (** `/>` **) used on a non-void HTML element. Ignoring the slash and treating as a start tag.**
   From line 7, column 5; to line 7, column 35
   `body>↩     <jsp:include page="menu.jsp" />↩ ↩`
4. **Error** **End tag for** `body` **seen, but there were unclosed elements.**
   From line 10, column 1; to line 10, column 7
   `ful!</h1>↩</body>↩</htm`
5. **Error** **Unclosed element** `jsp:include`.
   From line 7, column 5; to line 7, column 35