Why we chose AngularJS and Why 1.5.8?

We chose AngularJS,with its latest version 1.5.8, because it is the choice of most client side developers and also because of what it offers. To start if off:

1. Its really easy to **SET-UP**. Make the main module js file, add the ng-app="AppName" directive to your HTML, add the Angular library as a script tags dependency and you can start using angular data binding out of the box.
2. It offers great **HTML DOM Manipulation**. Most behaviors like defining onClick events, looping & class manipulation happen inside the HTML itself. Although angular offers a lot of directives. If I need a specific behavior i can make a custom directive that defines that behavior and add it to the html.
3. **2 way data binding** helps me bind model variables to html elements. This means when the model changes, the html element changes too. When I declare the ngModel directive to a html input element, I can dynamically change the value of the model and vice versa.
4. It also has some kind of **Dependency Injection**. When asking for specific dependencies, i can just add them as parameters to my controller.

One of the main reasons is because we have been using it for the past projects that requires fast paced development. Those are just a few reasons why we chose AngularJS. As what we have also mentioned during our chat, I am currently studying ReactJS but dont have much experience with it yet.

How does the UI communicate with the backend?
Or what is the relationship between UI and backend?

We use HTTP requests to communicate with the server. We used a core angular service ($http) which facilitates communication with our remote server. $http uses the browsers XmlHttpRequest object or via JSONP.

Why WebApp and WebService are on Google Cloud Platform?
Why the DB is on GoDaddy?

When we first searched for a server to host our application, GoDaddy was our first choice. We used their 1 month free subscription. The DB and the WebApp was also hosted there. When we hosted the WebService, we realized that we were getting errors due to the memory allocation of the vm. Since resizing the memory allocation means we'd have to avail of a new server and it means wed have to pay extra. So we opted to search for another free service so we ended up with google instead since it offered a 60 days free trial for its services. Lastly we decided to let the DB remain on the GoDaddy server so we could save time on setting it up on another server. That's basically all the reason to it.

What server specs did you use and why?

For the GoDaddy Server, we used a 512MB memory, 20GB SSD storage, 1CPU virtual maching running on cent-os 7. Since we are just using a free account, we didn't have much choice on customizing specs for the GoDaddy server.

For the Google Compute Engine Server, we were able to choose more freely on the specs to use. We went with a 7.5GB memory defaulted with a 10gb Standard Persistent Disk and 2CPU virtual machine running on Debian GNU/Linux 8.

Are you using some special features of Google Compute Engine? e.g., Auto scaling
If so what features and why?
If not why? And what would you use?

Sadly, No we didn't use auto scaling. In fact it is our first time personally setting up a server for hosting. We didn't have much experience in setting servers, since the servers we have been using was set up by someone else. But we are glad to say, we learned a lot just by doing this mini project.

What strategy are you using or would you use for DB backups?

Unfortunately we don't have much experience in DB management. So we can't yet say for sure what to use, but we are always willing to learn.

Monitoring?
Did you set up alerts for monitoring? Plugins? Services?
How would you go about availability and other non-functional requirements?

As we have known, and have done, we have setup Google Stackdriver for monitoring our projects. But we still don't have much experience in using it so we still don't know how to fully utilize it yet.

Why did you choose bcrypt as password encoder?

There are 2 main reasons why we used bcrypt as a password encoder:

1. It is fairly easy to setup in Spring, and is in fact spring also recommends it.
2. Bcrypt has been published and in use and probably been inspected for flaws by many people over quite some time.

In case you are using something to protect the service and web app from attacks, please explain, e.g., `iptables(8)`, CloudFlare

Unfortunately also, we don't have much experience in this aspect. So we can't say for sure.

How did you divide the work/tasks in this project?

Fealrone Alajas and Vincent Montesclaros worked together to make the WebServices.
Fealrone Alajas, was in charge of JUnit Testing.
Vincent Montesclaros, also made the Swagger Documentation/API.
The WebApp/User Interface was designed and developed by Donraedel P. Sumayan.
The servers was setup by Donraedel P. Sumayan.

> Although everything is basic, rest assured if given much more time, we can still improve this.
What things would you do for the next iteration?
And for the next after?

For the next iterations, we've envisioned our app to be some kind of Tasks/Team Management System with these following features:

1. A user can be part of a project/task team.
2. A user can be a project leader and add other users as members.
3. A project leader can add specific tasks to a team board.
4. A project leader can delegate tasks to a member.
5. A project leader can set task due dates.
6. A user can set tasks as ongoing,done,cancelled.

These are just a few improvements we have thought of.

In the technical side:

1. Improve on security.
2. Improve on error handling both on client and server side applications.