Josh Gardner

2/1/2024

CPSC 4911 Sprint 2 Notes

Please take notes on the technologies that your team has decided to research. There are some examples provided next to each prompt, but these lists are not exhaustive.

**Note: Due to the nature of some web-development frameworks (i.e., SvelteKit, Django, Next, …) your frontend and backend may be hosted by the same technology. In this case, please take notes on the specific parts of the framework in the sections below.

Frontend Language(s) (Javascript / Typescript / JSX / Svelte)

**Note: Backend languages such as PHP and Python will template/render some variation of JavaScript and HTML to display in the browser. If you are using a backend framework, recognize what language it will be rendering.

URL(s):

Notes:

- JS
    - Slightly limited compared to JSX when it comes to react
- JSX
    - Mix of JS and html
    - Is Javascript XML
    - This is probably the best language to use for the implementation we have
    - Extension of JS with HTML elements mixed in
    - Allows you to create interactive UI components easily

Frontend Framework/Architecture (Vue / React / Svelte[Kit] / Raw)

**Note: If using a backend framework with a raw frontend language please take notes on how the backend processes and renders front end code into a completed file (e.g., PHP will echo HTML from the server into the returned file)

URL(s): https://react.dev/

Notes:

- Key features include Hooks, Components, APIs, Directives
- Combination of HTML and JS

    - JSX
- Can be used for customer facing UI design
- Can handle interactive elements like maneuvering through webpage, calling external APIs for catalog, displaying data in visually appealing way (such as displaying a Sponsor's catalog)
- Focused on creating reusable UI components
- W3 Schools link: https://www.w3schools.com/REACT/DEFAULT.ASP
- Super modular and easy to work with
- Very popular framework in industry

Back End Language (PHP, Javascript, Python)

URL(s):

Notes:

- **Javascript**
    - Would be used with Node.js
    - Has the potential to also be used for front end, creating a more streamlined development
    - Used for full stack
    - Not too difficult to learn
    - Lots of documentation
- **Python**
    - Would be used with Flask
    - Cannot be used with frontend though
- **PHP**
    - Group has some experience with this language
    - Not used with Node or really meant for flask

Back End Framework/Architecture (Laravel, Django, Flask, Node.js)

**Note: Though it is not usually recommended and is often harder, you may choose to use the raw version of some of these languages. As a general recommendation you should look to use a trusted framework for efficiency and security reasons.

URL(s): EC2 Research:
https://docs.aws.amazon.com/ec2/?icmpid=docs_homepage_featuredsvcs

Notes:

- Flask
    - Better for python
    - More flexible and simple
    - Microframework
    - Default server is synchronous
    - Not as efficient at handling concurrency
    - Not as good with I/O bound applications
- **Node.js**
    - Asynchronous and event driven
    - Can improve performance and scalability
    - Great for high concurrency like handling multiple sponsor and drivers interactions
    - Javascript could be used for front end and back end, simplifying learning load
    - Great libraries for development
    - Non blocking I/O
    - Pyramid of doom
        - Callbacks nested in callbacks can create lots of complexity
        - Is avoidable with newer features
    - Seems to be better suited for this project

EC2

- Enforce standard naming convention with Team29
    - Make sure it is descriptive for the specific service
- AMI
    - Template that contains OS, application server, and applications which are included when server is launched
    - Amazon Linux tends to work best with free tier
        - This is the only AMI which is officially supported and maintained by amazon
        - Optimized for ec2 with ec2 api tools
        - https://ec2-downloads.s3.amazonaws.com/AmazonLinuxAMIUserGuide.pdf
- Discuss key pair login information and how this will be shared/stored for each of us
- Security group configurations
    - Where will we allow ssh traffic from
    - HTTPS
    - HTTP

- Do we want to establish IAM profile before creating server
- Should enable termination protection to prevent accidental deletion
- Brainstorm any user data we want this server to start with
    - Could use this for server to come with apache when started but this is minor

Relational Databases (PostgreSQL, MySQL)

**Note: Teams may find views and triggers to be beneficial as well as standardize operational queries. Please emphasize learning joins, grouping, ordering, and filtering. Finally, take notes on technologies/libraries in your backend language used to connect to the server.

URL(s): RDS research: https://docs.aws.amazon.com/rds/

Notes:

- our flexibility for RDS in this project comes in the following places
    - Master username/password
    - Engine type
        - We have chosen MySQL because of our familiarity with it
            - Joel has brought up MySQL plugin for VS Code
    - Connection to EC2 Compute resource
        - We need to discuss if this will be connected to our EC2 server
        - This is something that can be established after creation of the RDS server
    - VPC
        - Cannot be changed after creation of RDS
    - Public access
        - Going to be yes
    - Database authentication
        - Are we just doing password auth or are we including IAM database authentication
    - Enhanced monitoring
        - Provides additional metrics which are delivered to CloudWatch Logs
        - These metrics are displayed as graphs
        - Ask ‘Client’ about their opinion on this
        - Documentation: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Monitoring.OS.overview.html
    - Enabling encryption is something we want to do I assume
    - Also assuming we want to keep the port of 3306 the same

MySQL

- Can use MySQL Workbench or VS Code plugin to connect to RDS and create databases
- Need to think about SQL scripts needed to create appropriate DBs
- Map out different DBs, primary keys, and foreign keys for each
- Be intentional with planning to limit data redundancy
- Users table could have unique UserID as primary key, potentially having Username as foreign key
- Drivers table could use UserID for primary
    - Sponsor ID as foreign
    - Will sponsors assign Sponsored Drivers IDs to drivers?
- Sponsors will need to have their own Sponsor ID
    - Could be primary key for Sponsor table
- Orders table
    - Includes OrderID as primary key
    - Driver ID for foreign key
        - This is the driver who submitted the order
    - Potentially use Sponsor ID
        - For the sponsor who owns the catalog the order was submitted from
    -

****DEPENDENT ON SERVERLESS DECISIONS****

Web Servers (Apache, Nginx)

**Note: Emphasize research on permissions, services, virtual hosts, proxy passing and certifications.

URL(s):

Notes: