Hardware and software platforms:

Software -

What could be a constraint?
The product has to be built using Ruby.

How it could be a constraint?
Ruby is already present on most Macs but not Windows based PCs.

Why it is a problem?
If I were to move from working from Mac to my PC I'd need to install not only the correct version of Ruby to run this software but make sure all the gems I'm using are compatible with that version. The Ruby community is more geared towards dealing with Mac & Unix based users compared to Windows users so they're more likely to get newer better supported versions of Ruby.

Solution:
Don't upgrade my current version of Ruby on my Mac and avoid using my PC to develop the project to avoid version conflicts. If for whatever reason I had to change my device then by avoiding going after the very latest versions of Ruby on the Mac I'm more likely to find a version that's compatible with my PC.

Hardware -

What could be a constraint?
There's always the risk of dropping my laptop or knocking it against something damaging the hard disk.

How it could be a constraint?
Laptops aren't durable devices and since I'm working from home rather than an office with a large desk designated for work there's an increased risk of knocking my laptop off the table or dropping something heavy onto it by mistake.

Why it is a problem?
If my hard disk is badly damaged all the work I have for the project will be lost meaning I'd have to start again from scratch.

Solution:
Clear my work area of anything non essential materials for work and get a large enough desk to place my laptop reducing the risk of me dropping it or dropping something on it. Also commit my code regularly to GitHub so even if my hard disk gets damaged I still have access to some versions of my code online.

Performance Requirements

What could be a constraint?
My application needs to be scalable & flexible following MVC principals.

How it could be a constraint?
There is always the risk of me violating MVC principals such as putting too much logic inside my Views when trying to take short cuts.

Why it is a problem?
The Views are intended for displaying data to the end user whilst Modules are meant to handle the bulk of the business logic. If I add too much logic complexity to my Views I'll then find the code far harder to debug (especially since the Views not only contain Ruby but CSS & Hmtl), open

it up to greater security issues and look in the wrong places of where I'd expect to find things after returning to my code.

Solution:
Do my upmost to keep logic in my Modules and focus purely on the presentation of data in my Views. My project will be far more scalable should I decide one day to make radical changes to the format of my Views without wanting the change to impact my Modules. Also refactor my code to reduce any redundant functions in my Views seeing if they'd be better suited to either the Module or the Controller.

Persistent Storage and Transactions

What could be a constraint?
My app needs to be able to store user data and actions carried in it, such as which customer is assigned to why lesson and how that impact's the lesson's maximum capacity.

How it could be a constraint?
The owner of the app needs to be aware of the decisions they've made when it comes to allocating customers to lessons so the right customers go to the right lessons.

Why is it a problem?
If the data isn't stored somewhere the app owner won't be able to recall which customers should be in which lessons and their business will start to suffer. Especially if a customer is added to a lesson that's already at maximum capacity. The moment the app is turned off or the customer visits another web page to the one they were at previously the data will be lost.

Solution
Store all transactions in a PostgreSQL database with relevant tables so nothing gets lost if the app is turned off and on again. The app is too complicated to just store in the information in a simple text file.

Usability

What could be a constraint?
The expected client isn't a developer that's used to working with terminals or querying databases.

How it could be a constraint?
The client would expect the app to have user interfaces like web forms to carry out their transactions.

Why is it a problem?
The client would either get frustrated and give up using the app if its interface is off putting or waste time trying to learn an interface they were unfamiliar with. Also if the client whose unfamiliar with SQL tries to query the database there's a greater risk of them damaging the tables.

Solution:
Use Sinatra for the app's front end so the user will be able to interact with webforms, something they're more familiar with. Using forms will also limit the powers they have compared with directly interacting with the database. Then the information can be transferred from Sinatra to Ruby business logic Modules and into the database.

Budgets

What could be a constraint?
The project has no budget.

How it could be a constraint?
Having no budget means being unable to purchase expensive software and hardware.

Why is it a problem?
I am limited to using free software and the hardware currently available to me. Had I a larger budget I'd be able to speed up the development process using expensive IDEs that'd help me to plan out, write & debug code at a faster pace. I'd also be able to buy software that runs at higher speeds so waste less time waiting for the app to load and expensive design software intended to help me plan out the project.

Solution:
Make the most of open source software freely available to assist development. Star UML is open source and has helped me come up with UML diagrams, Pencil.app can help with non UML diagrams and VSCode is a free to use text editor that comes with plenty of helpful code snippets and formatter extensions.

Time Limitations

What could be a constraint?
I have 1 week to complete the project.

How could this be a constraint?
The project needs to be produced within a deadline or it will be considered a failure.

Why is it a problem?
With time constraints I need to make sure I dedicate adequate amounts of time to planning, coding and testing. The more time dedicated to one task means the others will suffer.

Solution:
Carry out the planning stage as soon as possible and try to predict problems I can run into and how to avoid them so less time is wasted during the coding stage so I'll then have time to properly test my code. Run some of my plans past my colleagues to see how realistic my ideas are and if they'll be achievable in the time limit. Make sure the MVP is delivery first and foremost so I don't get distracted by other tasks resulting in me failing to deliver the project on time.