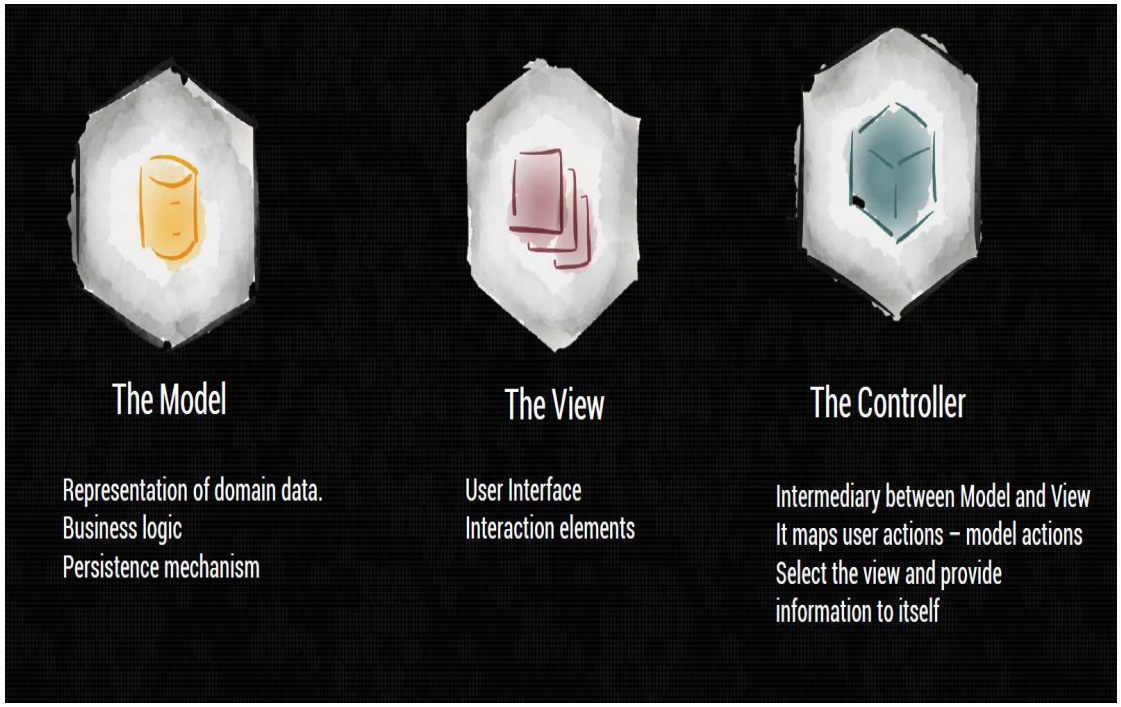
Design Pattern & Product Features

Week3 Class
Groups: L01A,L01B, L05A,L05B,L06A,L06B
Friday October 7, 2022

Design Pattern
MODEL VIEW CONTROLLER (MVC)

- Software architecture pattern that separates the model, the user interface and control logic of an application in three distinct components.
- MVC proposes the construction of three distinct components. One side for the representation of information, and on the other hand for user interaction.



MODEL

Which database table is the requested data stored in?

What SQL query will get me the data I need?

Talks to data source to retrieve and store data

VIEW

Would this text look better blue or red? In the bottom corner or front and center?

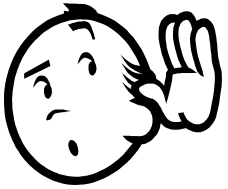
Should these items go in a dropdown list or radio buttons?

Asks model for data and presents it in a user-friendly format

CONTROLLER

The user just clicked the "hide details" button. I better tell the view.

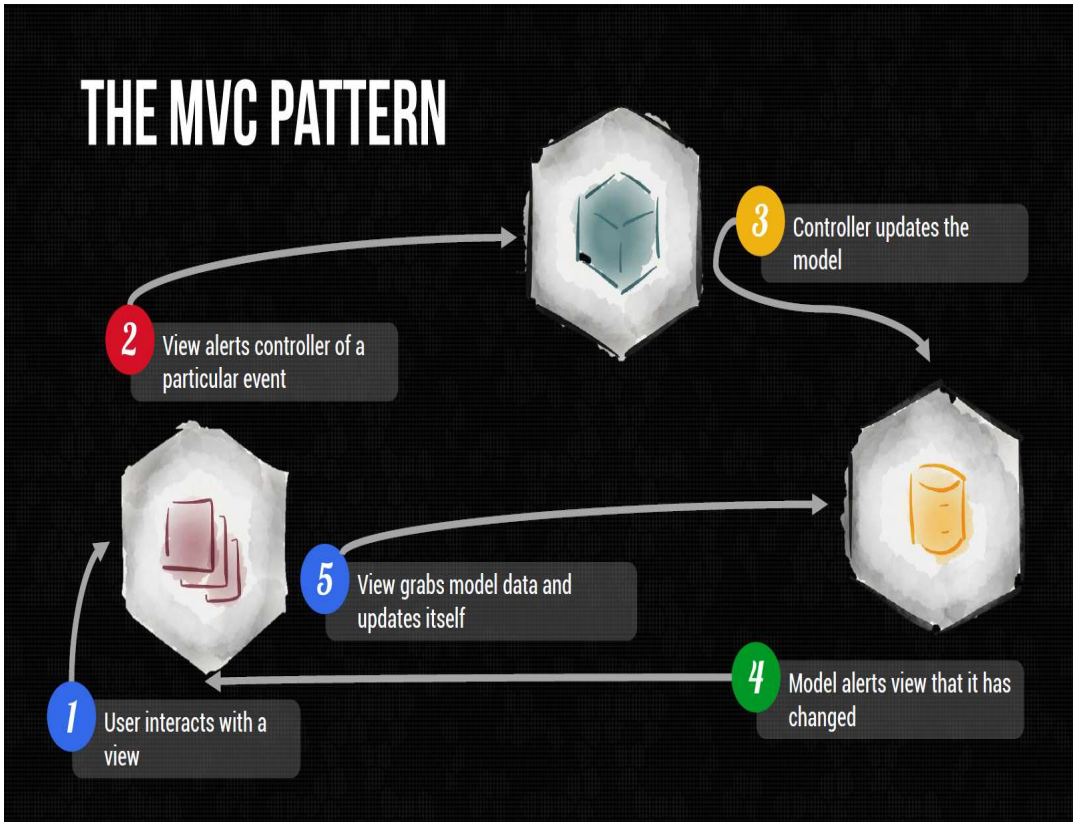
The user just changed the event details. I better let the model know to update the data.



Listens for the user to change data or state on the UI, notifying the model or view accordingly

MVC PATTERN

1. User interacts with a view
2. View alerts controller of a particular event
3. Controller updates the model
4. Model alerts view that it has changed
5. View grabs model data and updates itself

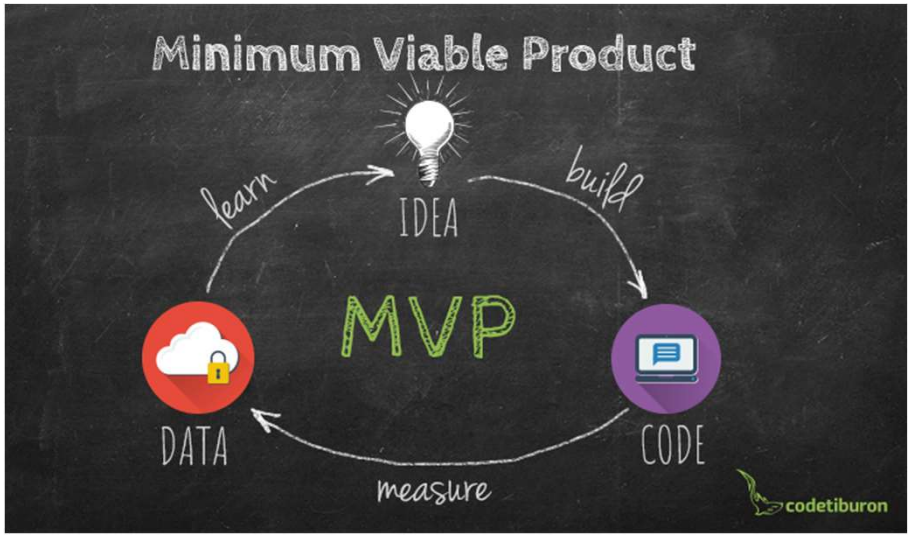


Benefits of MVC

- Organization of code
  - ❖ Maintainable, easy to find what you need
- Ease of development
  - ❖ Build and test components independently
- Flexibility
  - ❖ Swap out views for different presentations of the same data (ex: calendar daily, weekly, or monthly view)
  - ❖ Swap out models to change data storage without affecting user

Minimum Viable Product (MVP)

- A product with enough features to validate a product idea early and attract early-adopter customers/stakeholders
  - Receive user feedback as quick as possible
  - Improving product by iterating quicker
  - Learn what does/doesn't work for users



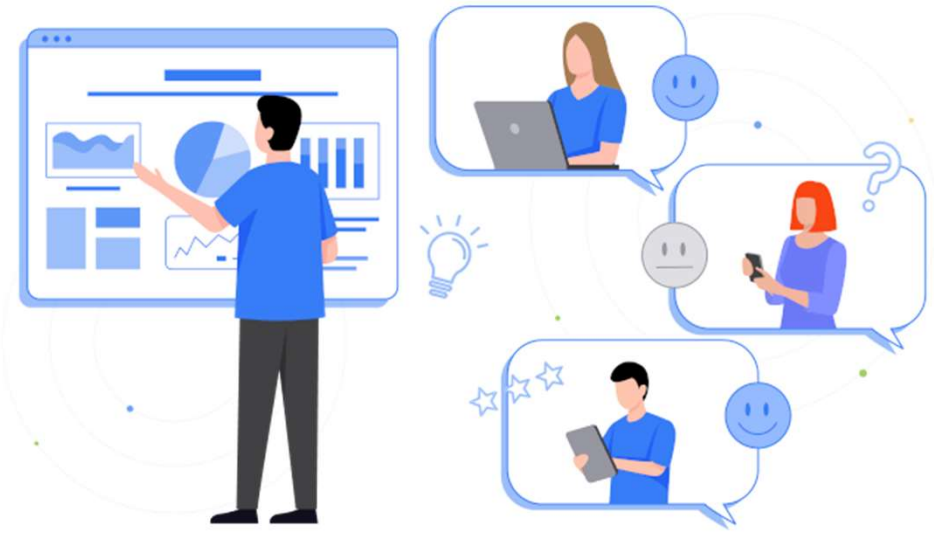
How do we define the MVP?

1. Ensure MVP aligns with business objectives.
2. Identify specific problems you want to solve or improvements you want to enable for user persona.
3. Translate your MVP functionality into a plan of development action.

Example(s):

Growing Past the MVP

1. Have I gathered customer feedback?
2. Do I know who my customer is?
3. Do I know what I need to improve?
4. Is this product worth it?

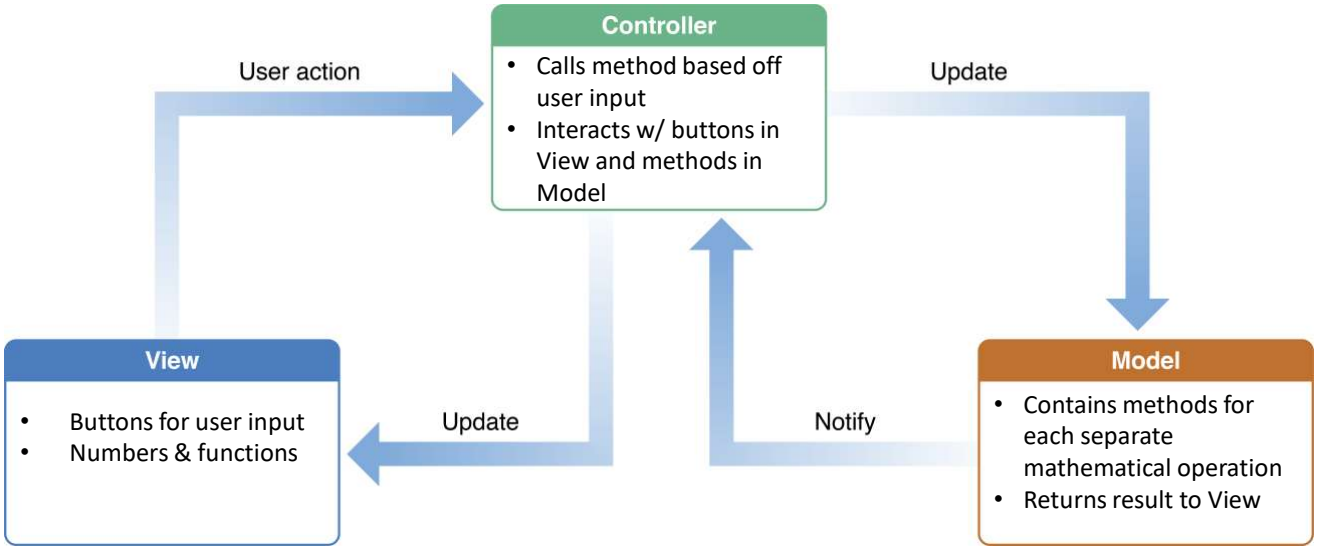


Example: Calculator Application

- Feature(s):
  - Add/Subtract/Multiply/Division
  - Log/Exp/Factorials/etc
  - UI Interface
  - Graphing

How can we use the MVC pattern to approach a calculator application?

What should be included in our MVP for such an application?
- Minimal Operations
- UI Interface



As you create your mock-up(s):

Consider:

1. How can you utilize the MVC pattern to help architecture your budget management platform?
2. What problems are you trying to solve with your budget management platform?
3. What feature(s) should be prioritized and included first?