TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

Software Engineering Design Principles

Model-View-Controller

PRESENTED BY:

Model–View–Controller (MVC)

- *Architectural and Design pattern* (i.e. reusable solution)
- Used for developing user interfaces that divides an application into three interconnected parts.
- Why?
  - Separate internal representations of information from the ways information is presented to and accepted from the user.
  - *Design pattern* decouples these major components allowing for efficient code reuse and parallel development.

Architectural Pattern

- General reusable solution to a commonly occurring problem in software architecture within a given context. Architectural patterns are similar to software design patterns but have a broader scope.
- Examples
  - ETL (data extraction transformation and loading)
  - Data warehouse
  - Data science and advanced analytics
  - E-R data modeling
  - Deep learning
  - MVC

Design Patterns

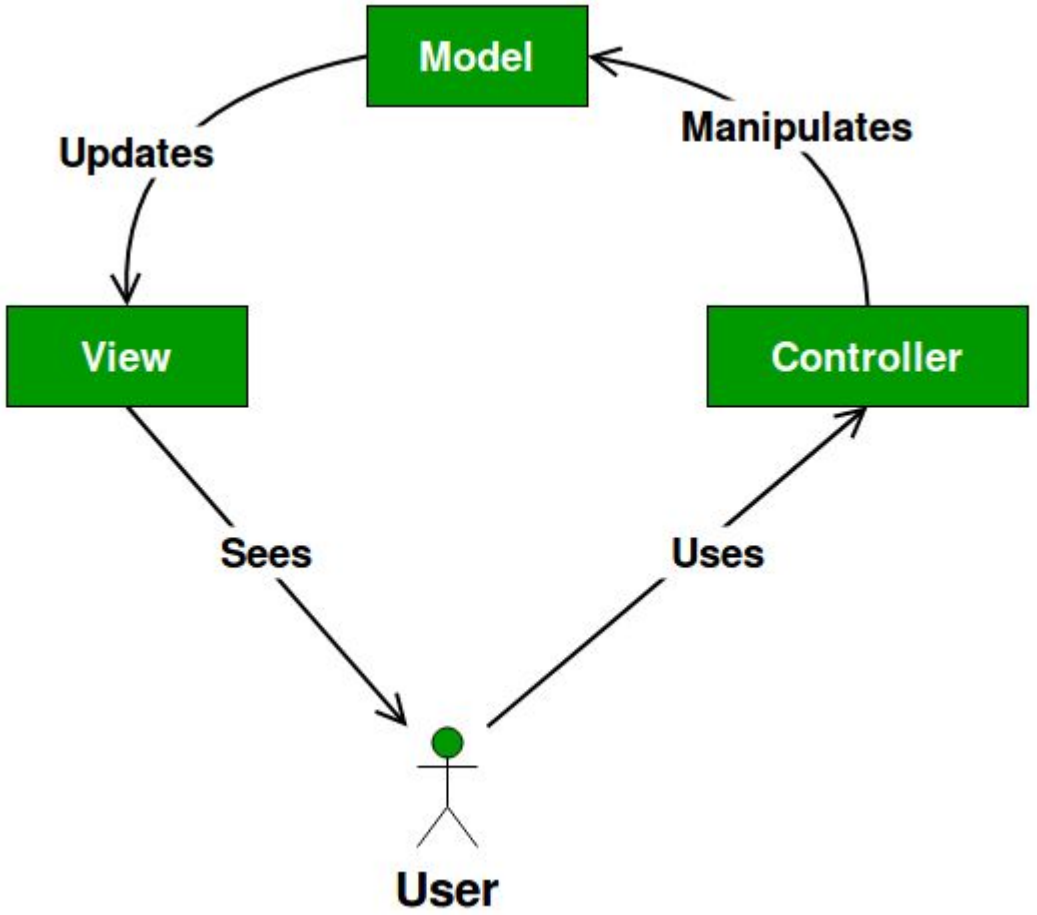
- The Gang of Four are the four authors of the book, "Design Patterns: Elements of Reusable Object-Oriented Software".
- Design patterns provide solutions to common software design problems - generalised solutions in the form of templates that may be applied to real-world problems.
- MVC as a Design Pattern
  - MVC: One Model, One view, the Controller manages the communication between them.
  - Observer Pattern: One Model, Multiples views (observers/subscribers), and the publisher manages the communication

More Why

- Simultaneous development
  - MVC decouples the various components of an application
  - Developers are able to work in parallel on different components without impacting or blocking one another. For example, a team might divide their developers between the front-end and the back-end.
- Code reuse
  - Developers are able to reuse components quickly and easily in other applications.
  - The same view for one application can be refactored for another application with different data because the view is simply handling how the data is being displayed to the user.

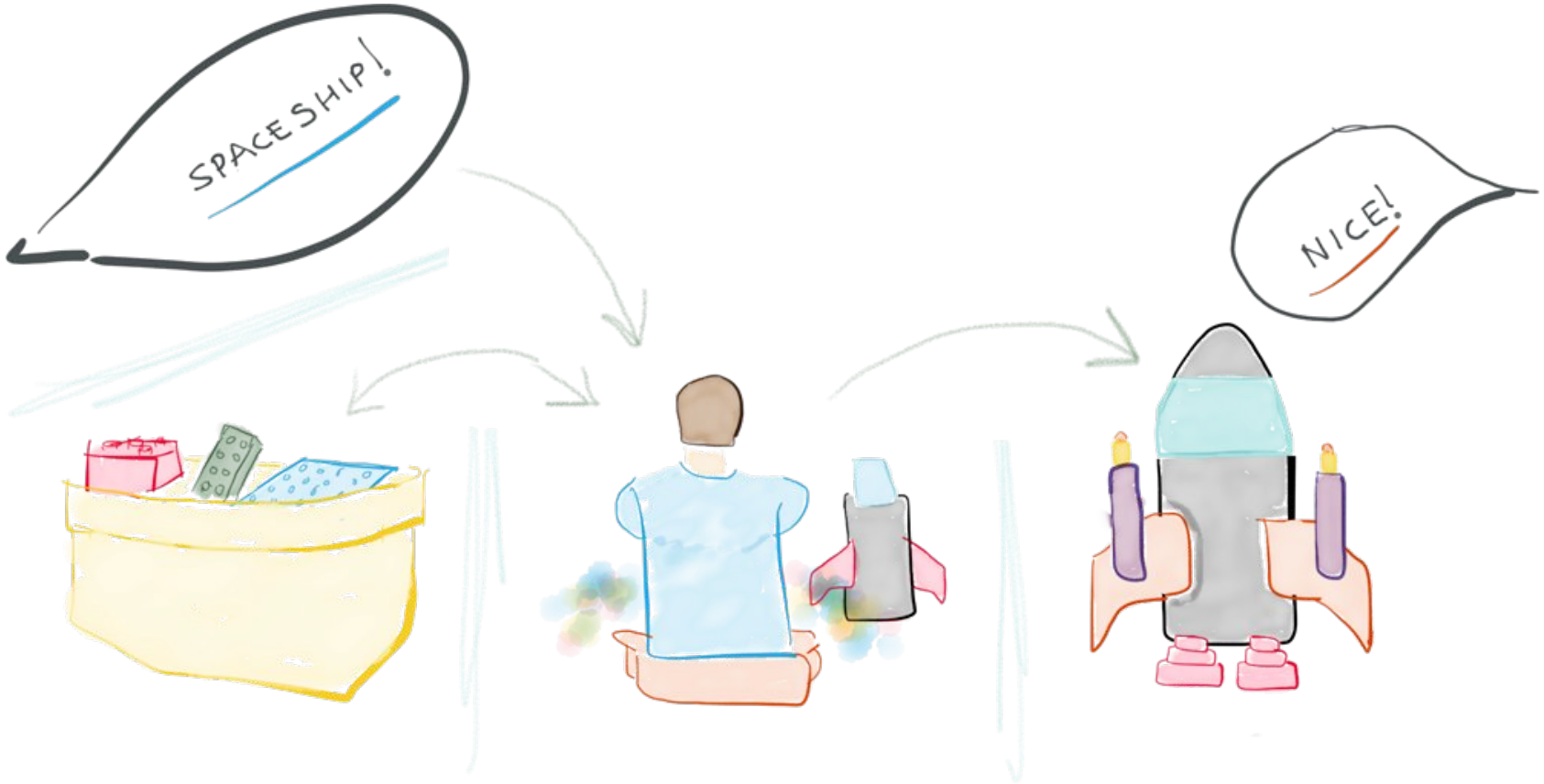
Components

- **Model** - The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.
- **View** - Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
- **Controller** - Accepts input and converts it to commands for the model or view.
- **Interactions**
  - The model is responsible for managing the data of the application. It receives user input from the controller.
  - The view means presentation of the model in a particular format.
  - The controller responds to the user input and performs interactions on the data model objects. The controller receives the input, optionally validates it and then passes the input to the model.



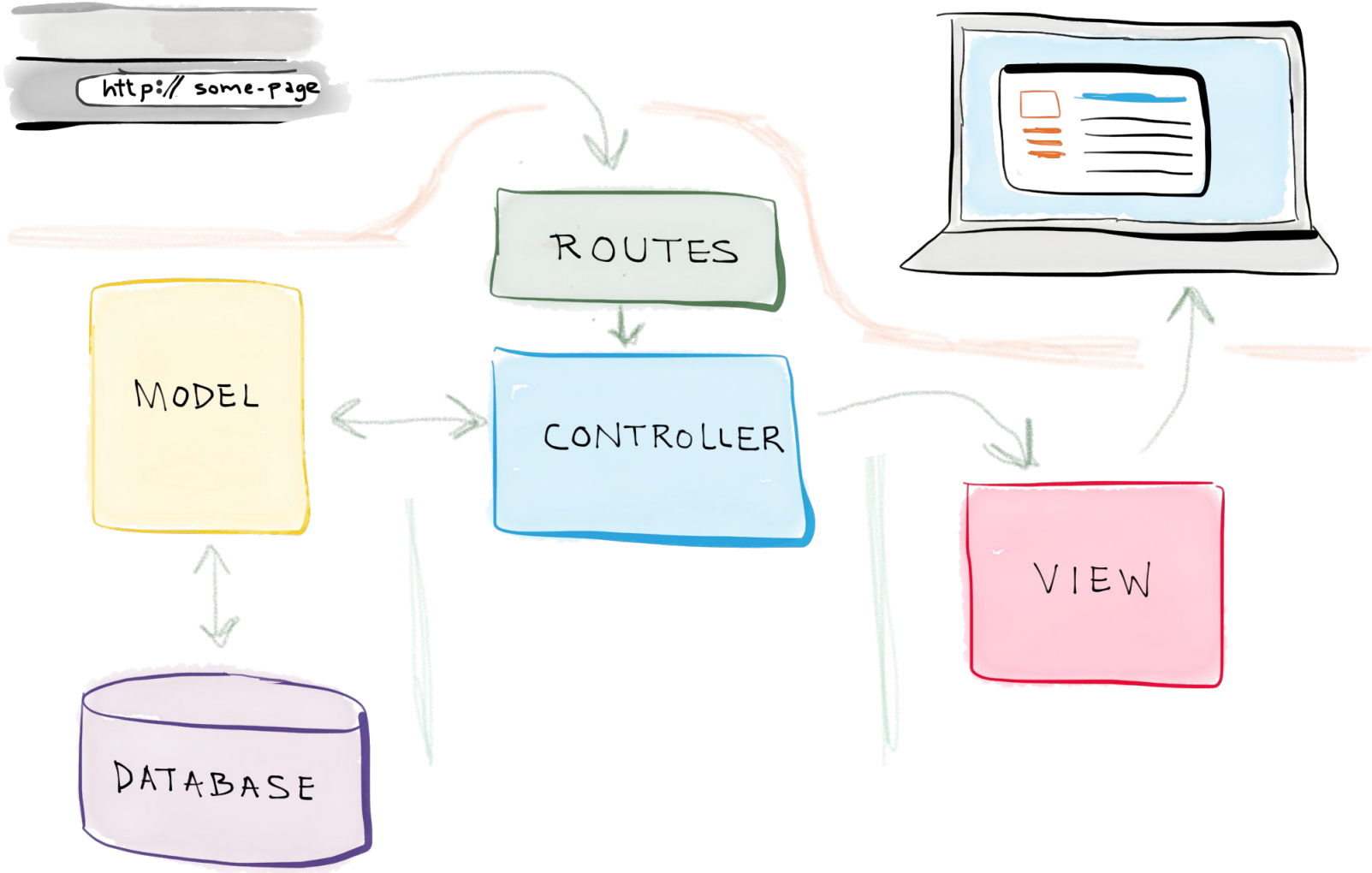
Legos Example

1. Your brother makes a request that you build a spaceship.
2. You receive the request.
3. You retrieve and organize all the Legos you need to construct the spaceship.
4. You use the Legos to build the spaceship and present the finished spaceship back to your brother.



Web App Example

1. A user requests to view a page by entering a URL.
2. The Controller receives that request.
3. It uses the Models to retrieve all of the necessary data, organizes it, and sends it off to the…
4. View, which then uses that data to render the final webpage presented to the the user in their browser.



Flask app

- Routes
  - Each route is associated with a controller or a function within a controller, known as a controller action. So when you enter a URL, the application attempts to find a matching route, and, if it's successful, it calls that route's associated controller action.
- Model - Controller
  - The controller uses the models to retrieve all of the necessary data from a database; and that data is passed to a view, which renders the requested page. The data retrieved via the models is generally added to a data structure (like a list or dictionary), and that structure is what's sent to the view.
- Views
  - The structured data that the end user sees
  - Flask app, we can loop through the entries, displaying each one using for example JSON or the **Jinja** syntax:

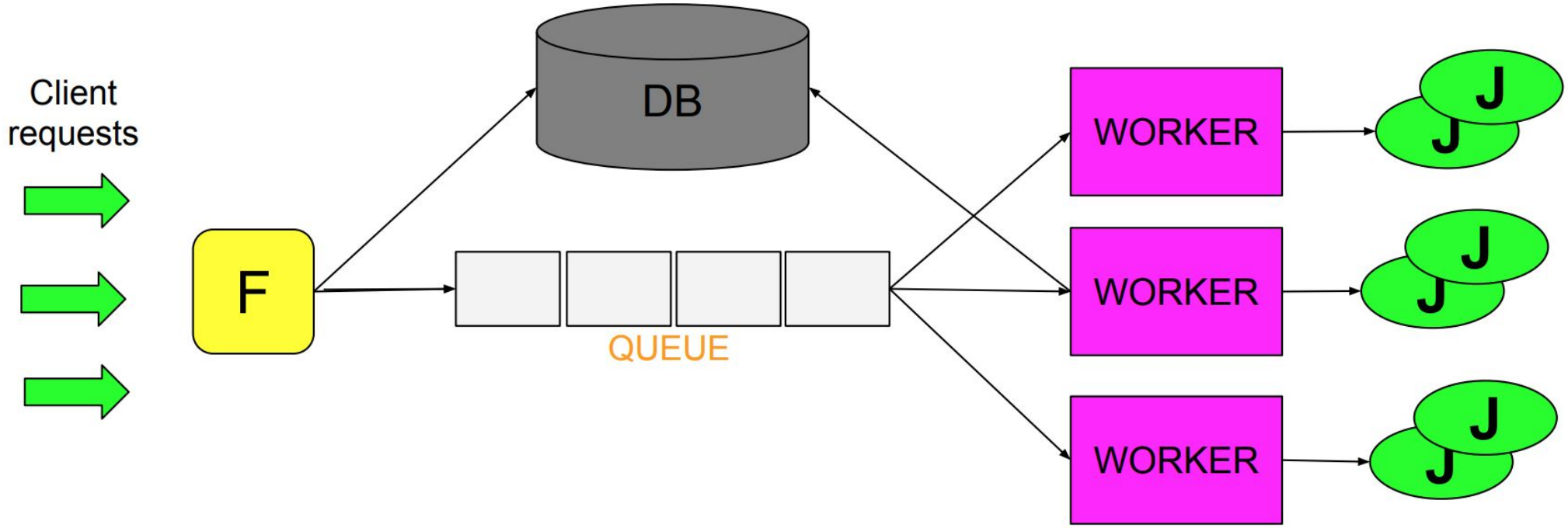
HTML

```
{% for entry in entries %}
  <li>
    <h2>{{ entry.title }}</h2>
    <div>{{ entry.text|safe }}</div>
  </li>
{% else %}
  <li><em>No entries yet. Add some!</em></li>
{% endfor %}
```

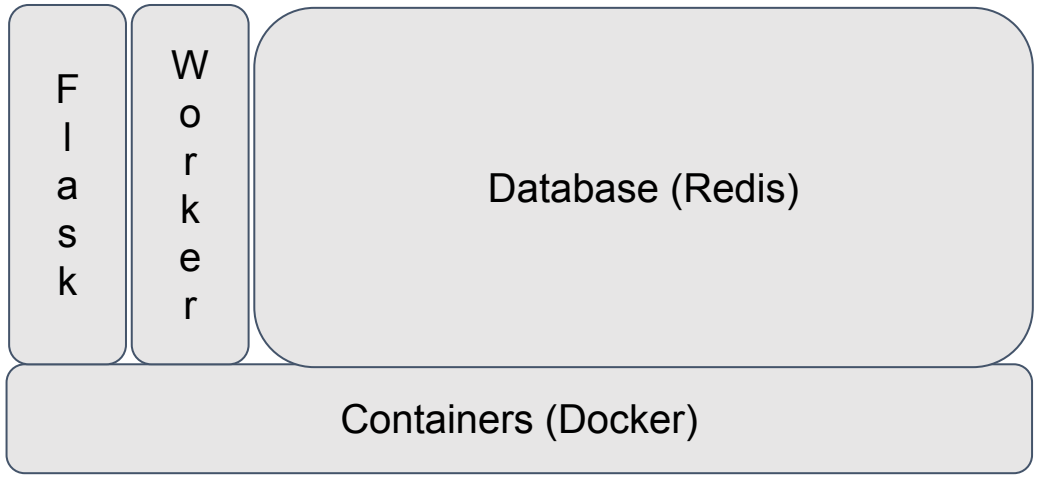
Examples of MVC Frameworks

- Flask
  - Lightweight - and **very** flexible. A lot more coding.
- Django
  - Forces rigid app setup in what can commonly be an hour-long setup or greater. But, a lot less to code!

Software Engineering, Basic Framework



Software Engineering + Docker Paradigm



Software Engineering + Docker Paradigm

Containerized Microservices

Highly maintainable and testable
Loosely coupled
Independently deployable
Organized around business capabilities