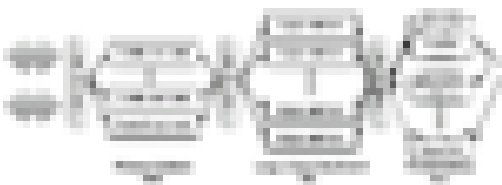
"Plan and Document": Before coding, project managers make plans. Write detailed documentation of phases of plan, things we can write against the plan. Changes to project cannot be reflected in the documentation and possibly to plan.

	Waterfall	Spiral	Agile
Description	1. Requirements analysis & specification 2. Architectural design 3. Implementation & Integration 4. Verification 5. Operation & Maintenance Depends on project manager	Combine Plan and Document with prototypes. Rather than plan & document all requirements 1st, develop plan & requirements documents across each iteration & prototype is created and review with the project	Document free zones, Agile emphasizes Test Driven Development (TDD) (write a number, receive down Time Release to validate customer requirements. Value by (avg total points/week) to measure progress
Benefits	Good for bigger teams	Involve prototypes & validations	Shorter iterations

HTTP

- HTTP request has header request method (GET, POST, etc.), Uniform Resource Identifier (URI), HTTP protocol version understood by the client, headers — extra info regarding transfer request
- Since HTTP is stateless, you use cookies in order to share data to help guide users through the pages of the app (e.g. authentication, favorite list tracking, recommendations, etc.)

3-Tier Shared Architecture



Shared Hosting Architecture

MVC

- Controller coordinates between models and views
- Any instance variable declared in the controller is made available to the view
- Ruby keeps Unix A controller and gives each model the ability to perform CRUD
- Any instance variable declared in the model that is not part of the database will need to be made available initialized for every request made to the app
- In MVC, each interaction the user can do is handled by a controller action
- Routes: routes maps <HTTP verb, URI> → controller action

Ruby

- Define time, lets us ask an object questions about what it knows, it usually itself
- Mixing/Mixing mixing, lets us define new role at runtime
- Modules = Collection of methods that aren't a class
- Ancestors is the set of all variable bindings you can see "now" at a given point in time. It's "I know every" all the variables you can see, including variables of variables
- Ruby blocks are closures: they carry their environment around with them
  - Block blocks can help you by separating what to do from when & where to do it
- Duck typing, you manage behavior, ensure "looks like" a method and ask "everything is a method, so do you to respond to this method?"