Joining Your First Collaborative Software Project

George Dill

Engineer / Founder Avondalien Software LLC

Avondalien
Software

Summary

- Goals change when you transition from student to collaborator
- Teams can decide what works best for them, but an agile framework can be a good way to collaborate
- Many popular tools exist to facilitate collaboration - we'll examine git and GitHub
- Software composition patterns that may seem onerous in solo dev actually facilitate group work

About Me

George Dill | Age 40

- Founder Avondalien Software
- Software Engineer since 2020
- Worked in enterprise shops like John Deere and Echo Global Logistics
- Earned a Masters Degree in CS from University of Illinois at Chicago
- Previously worked in Construction Mega-Project Management both in the US and South Korea
- Also ask me about AWS, React, and Pizza

Transitioning to Collaborative Projects

A Student’s Journey

George’s Perspective (USA)

Graduate

Work at an Internship

Study for Finals

Do a Group Project

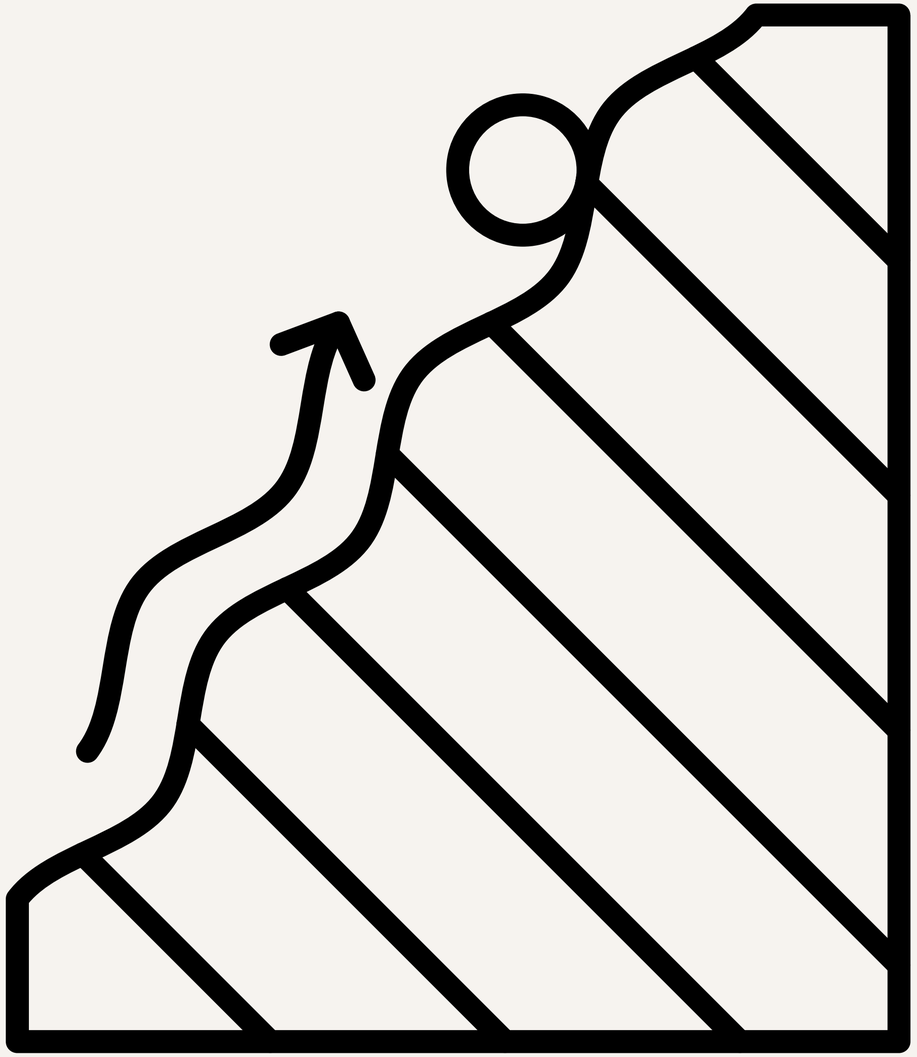
Study for Midterms

Get Admitted to a University

Take college admissions exams

Study flash cards

Learn to read



A Student's Journey

George's Perspective (USA)

Graduate

Work at an Internship

Study for Finals

Do a Group Project

Study for Midterms

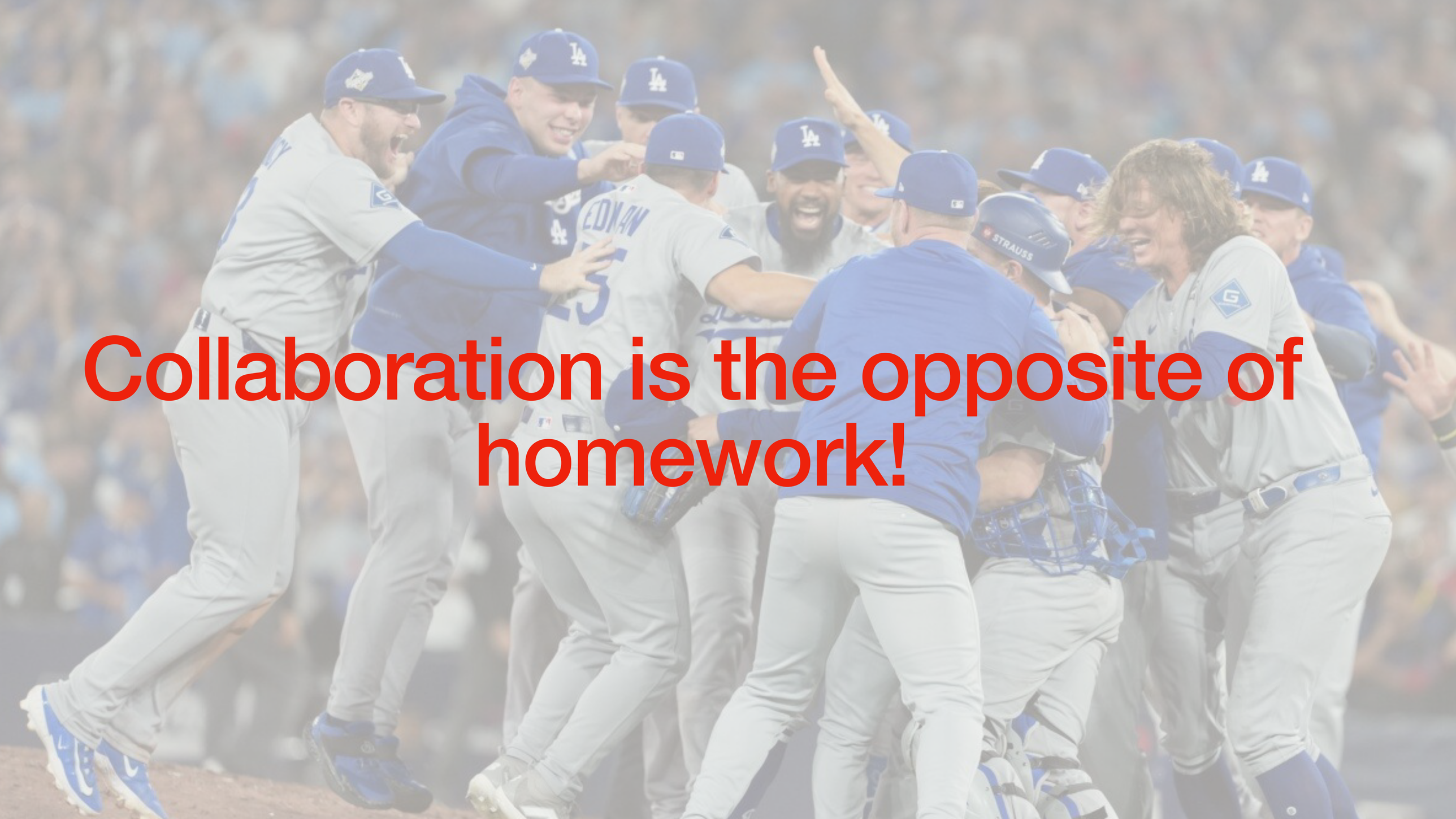
Get Admitted to a University

Take college admissions exams

Study flash cards

Learn to read

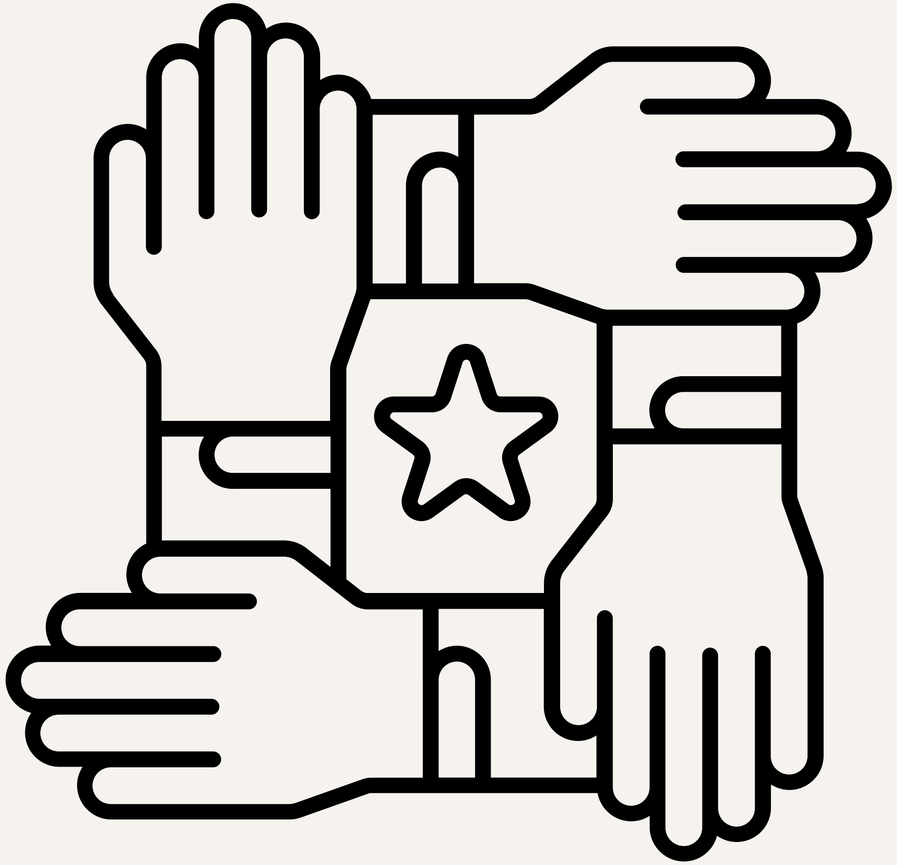
The student's journey is focused on individual mastery of course material



Collaboration is the opposite of homework!

Adopting a Collaborative Mindset

- Be vulnerable - Your teammates want to help you accomplish your goals
- Over communicate your progress and updates - You might think you're on the right track, but maybe you aren't.
- Help your teammates - when your teammate is on the group chat looking for help be there for them
- Always be learning - You'll pick up more knowledge by taking on a challenging task.



How Software Project Teams Work Together

Introduction to Agile

agilemanifesto.org

- **Individuals and Interactions** over processes and tools
- **Working Software** over comprehensive documentation
- **Customer Collaboration** over contract negotiation
- **Responding to Change** over following a plan

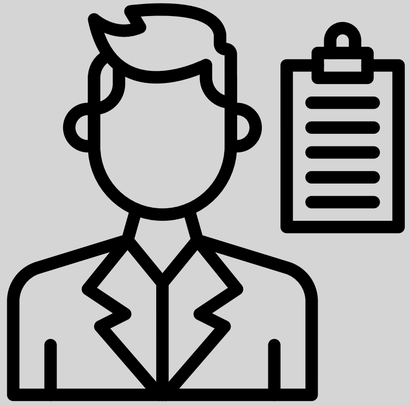
In Practice - A Typical Software Team

Product Owner

- Voice of the customer
- Makes decisions on software requirements
- Provides feedback on in-progress software
- Accepts the “finished” product

Engineers

- Architect a solution that meets the PO’s requirements
- Test their own work
- Provide value recommendations
- Implement the Project



Engineering Manager

- Provides guidance on software architecture
- Communicates project status to stakeholders
- Coaches engineers on opportunities for growth

An Engineer’s Day

In Corporate America

Task	Hours	Frequency
Coordinate with teammates	0.5	Daily
Clarify business requirements	1-2	Weekly
Architect a Feature - Get consensus on scope	2-4	Weekly
Review Code	2	Daily
Write Tests	2	Daily
Write Code	2	Daily
Demonstrate work in Progress	1	Weekly

How Do I Communicate?

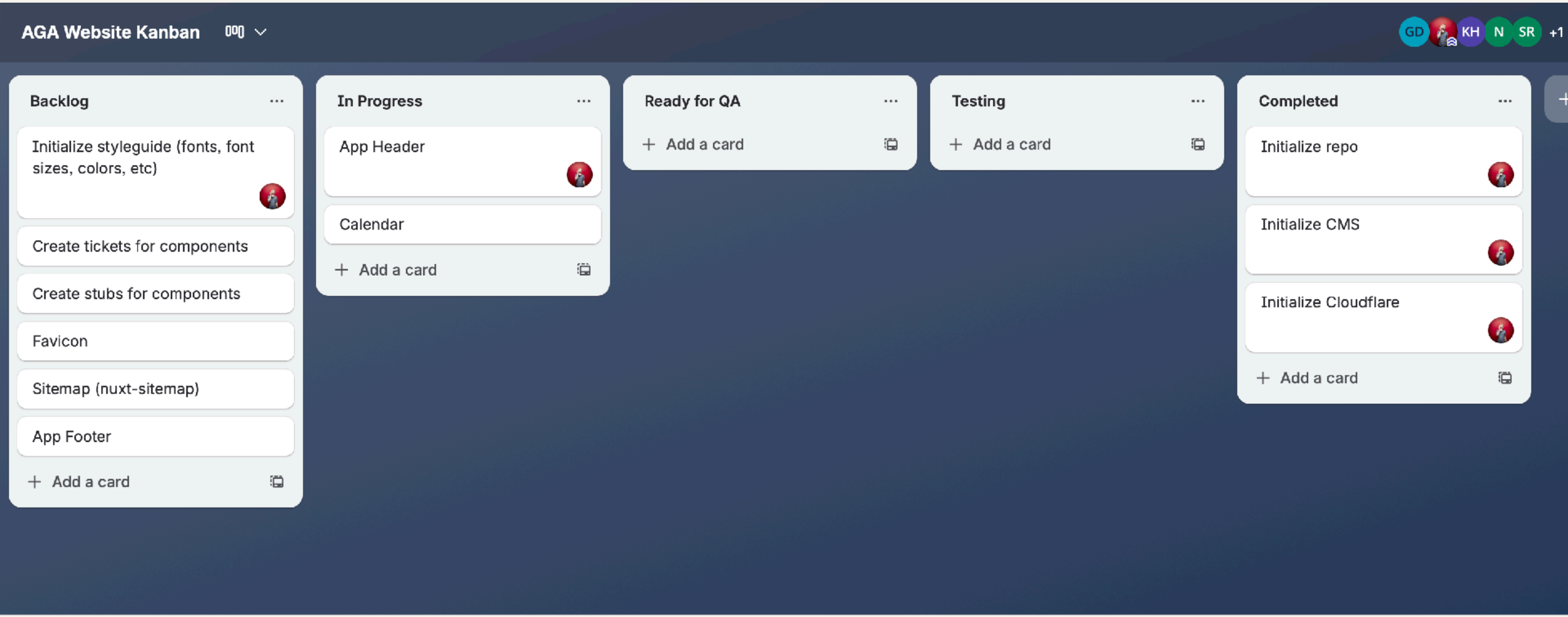
Strategies for Working Together

- Pair Programming - Look over your partner’s shoulder and talk it out
- Pull Requests - Work asynchronously and solicit feedback through an online form
- Daily Meetings - Short, regular get togethers to coordinate with your team on a regular cadence
- Messaging Apps - Find out which one your team uses and be active on it!

Collaboration Tools

What Should I Be Working On?

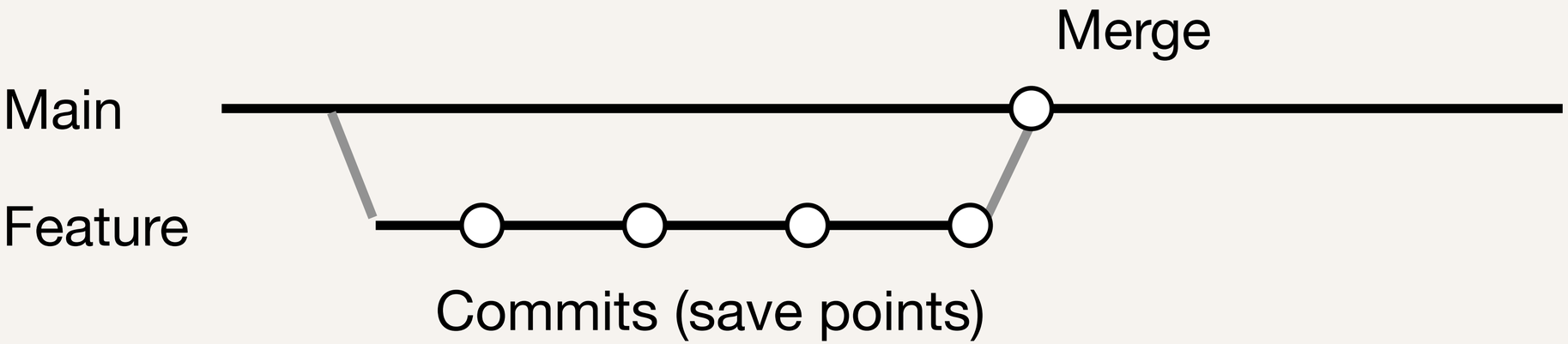
Kanban Board



How Do I Know My Code Is Current?

Version Control Systems and Trunk Based Development

- Git is a version control tool
- It allows you to collaborate with other authors in the same codebase using branching
- You can also create your own code save points using commits



How Do I Get Approval or Provide Feedback?

Git Hosting and Collaboration Platforms

- Provide cloud hosting for your code
- Collaboration tools like pull requests to easily analyze changes in code
- Can run automations based on check in actions to accomplish
  - Testing
  - Deployments
  - Security vulnerability scans

GitLab

Bitbucket

Github

Demo

1. Create a new branch with a descriptive name.

```
avondalien — -zsh — 80×24
gdill@macbookpro avondalien % git checkout -b feature/add-slide-deck
Switched to a new branch 'feature/add-slide-deck'
gdill@macbookpro avondalien %
```

2. Let your reviewer know you've started work - and what you plan to do!

```
gdill@macbookpro avondalien % git status
On branch feature/add-slide-deck
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/pages/home/OfferingCard.tsx
        modified:   src/pages/home/Services.tsx

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        src/assets/collaborative_software_development.pdf

no changes added to commit (use "git add" and/or "git commit -a")
gdill@macbookpro avondalien % git add -u
gdill@macbookpro avondalien % git add -A
gdill@macbookpro avondalien % git status
On branch feature/add-slide-deck
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   src/assets/collaborative_software_development.pdf
        modified:   src/pages/home/OfferingCard.tsx
        modified:   src/pages/home/Services.tsx
```

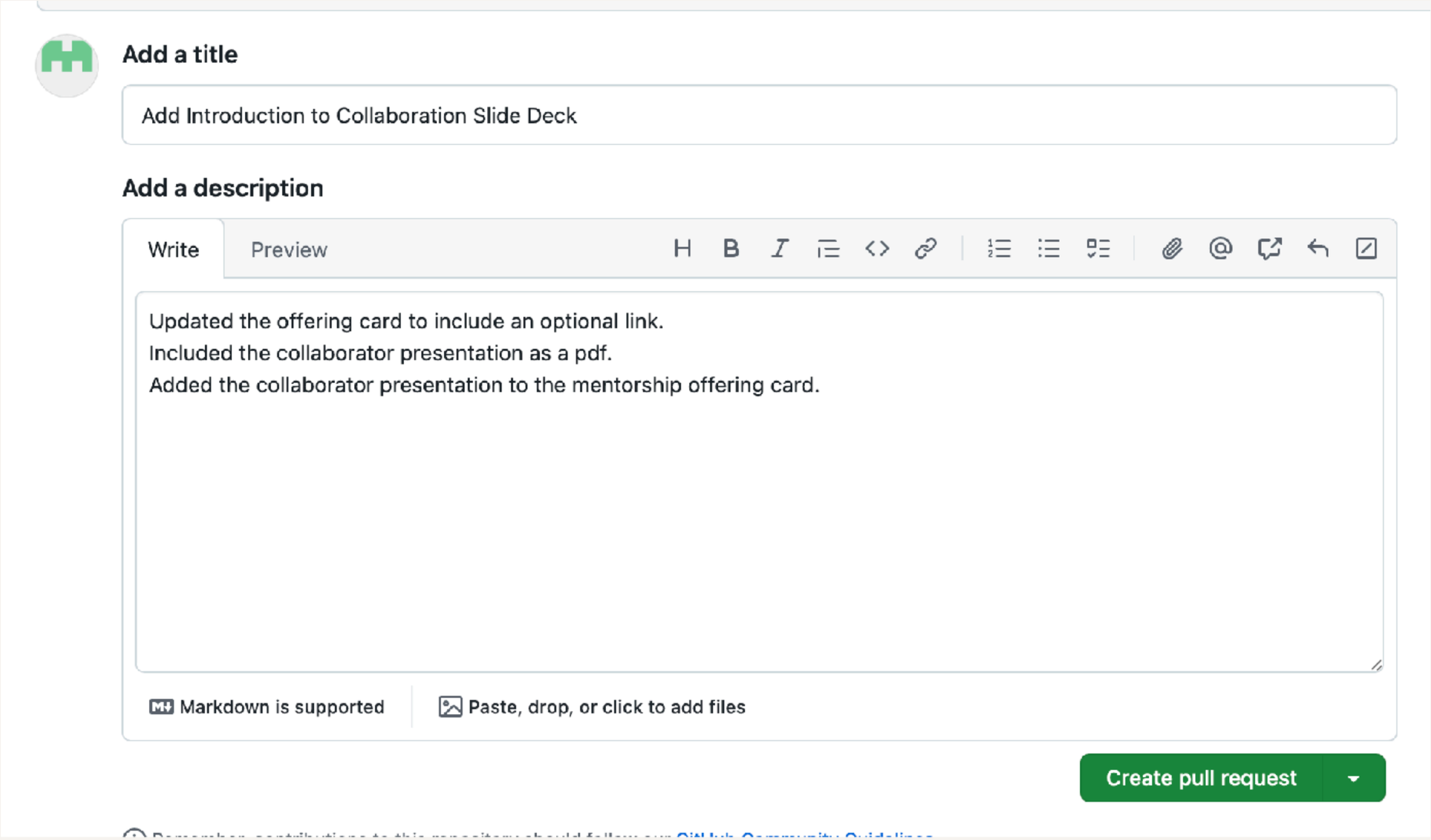
3. Make Changes

4. Check your changes into version control

```
gdill@macbookpro avondalien % git push --set-upstream origin feature/add-slide-d
eck
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 2.30 MiB | 6.49 MiB/s, done.
Total 9 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
remote:
remote: Create a pull request for 'feature/add-slide-deck' on GitHub by visiting
:
remote:      https://github.com/avondalien/homepage/pull/new/feature/add-slide-d
eck
remote:
```

5. Create a pull request on your git hosting platform

6. Send a link to the PR to your reviewer



7. Your reviewer looks over your work and can provide comments

Conversation 0 | Commits 1 | Checks 0 | Files changed 3

Changes from all commits | File filter | Conversations

Filter changed files

- src
  - assets
    - collaborative_software_dev...
  - pages/home
    - OfferingCard.tsx
    - Services.tsx

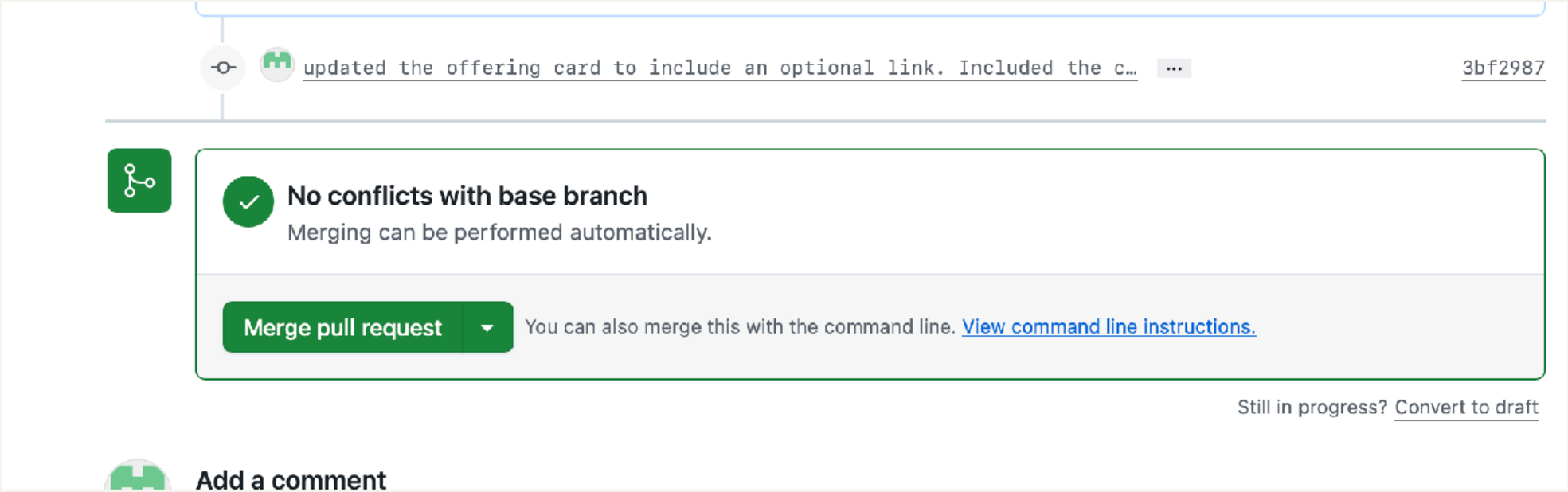
BIN +2.33 MB src/assets/collaborative_software_development.pdf

Binary file not shown.

12 src/pages/home/OfferingCard.tsx

```
@@ -10,14 +10,18 @@ interface OfferingCardProps extends PropsWithChildren{
10 10      description: string
11 11      image: string
12 12      imageAltText: string
   13 +    linkText?: string
   14 +    linkTarget?: string
13 15  }
14 16
15 17  const OfferingCard = (props: OfferingCardProps) => {
16 18      const {
17 19          title,
18 20          description,
19 21          image,
20    -        imageAltText
   22 +        imageAltText,
   23 +        linkText,
   24 +        linkTarget
21 25      } = {...props}
22 26
```

8. Once approved merge your changes by your teams release process



9. Automations will run. In this case the presentation has now been posted to avondalien.com

Software Patterns for Collaborative Projects

SOLID Principles

Guidelines for Writing Easily Reviewable and Testable Code

- Single Responsibility - Every unit of software should do just one thing
- Open-Closed - Your units of code shouldn’t be modified after they are completed, extend them instead.
- Liskov Substitution - Sub classes that inherit from superclasses should be replaceable by the super class.
- Interface Segregation - Interfaces should be as granular as practical so that implementers aren’t required to implement unused methods
- Dependency Inversion - Instantiate service classes and pass to their clients through the constructor to avoid deep coupling and make testing easier.

Test Driven Development

Try Writing Your Tests First

- Helps to reveal when dependency inversion and single responsibility principle should be applied.
- Helps reveal when interfaces might be a helpful construct
- Debugging with inductive reasoning - its easier to isolate problems when you know the building blocks are correct
- A well-written suite of tests can block errant commits from making it to production

Dependency Inversion - Example

```java
import com.DivvyAnalysisCompany.UntestableCode.Models.TripsByStationPairDateResponse;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.Errors;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.net.URI;
import java.util.Date;
import java.util.GregorianCalendar;

@RestController
public class TripCountAPI {

    @GetMapping(path="/TripsByStationPairDate", consumes="application/json")
    ResponseEntity<Object> GetTripsByStationPairDate(
            @Valid @RequestBody TripsByStationPairDateRequest request, Errors errors) {
        if (errors.hasErrors()){
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(errors.getAllErrors());
        }

        if (request.date.before(new GregorianCalendar(2013, 06, 28).getTime())){
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("{ error: Invalid date before service start date");
        }

        // Validate Station Identifiers
        StationDataProvider sdp = new StationDataProvider();

        if(!sdp.isValidStation(request.sourceStation)) {
            String responseMessage = String.format("Invalid station %s", request.sourceStation);
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body(responseMessage);
        }
        if(!sdp.isValidStation(request.destinationStation)) {
            String responseMessage = String.format("Invalid station %s", request.destinationStation);
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body(responseMessage);
        }

        // Get the Trip Count
        Integer tripCount = sdp.getTripsByStationPairAndDate(request.sourceStation,
                request.destinationStation,
                request.date);
```

Dependency Inversion - Example

```java
import com.DivvyAnalysisCompany.UntestableCode.Models.TripsByStationPairDateResponse;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.Errors;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.net.URI;
import java.util.Date;
import java.util.GregorianCalendar;

@RestController
public class TripCountAPI {

    @GetMapping(path="/TripsByStationPairDate", consumes="application/json")
    ResponseEntity<Object> GetTripsByStationPairDate(
            @Valid @RequestBody TripsByStationPairDateRequest request, Errors errors) {
        if (errors.hasErrors()){
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(errors.getAllErrors());
        }

        if (request.date.before(new GregorianCalendar(2013, 06, 28).getTime())){
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("{ error: Invalid date before service start date");
        }

        // Validate Station Identifiers
        StationDataProvider sdp = new StationDataProvider();

        if(!sdp.isValidStation(request.sourceStation)) {
            String responseMessage = String.format("Invalid station %s", request.sourceStation);
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body(responseMessage);
        }
        if(!sdp.isValidStation(request.destinationStation)) {
            String responseMessage = String.format("Invalid station %s", request.destinationStation);
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body(responseMessage);
        }

        // Get the Trip Count
        Integer tripCount = sdp.getTripsByStationPairAndDate(request.sourceStation,
                request.destinationStation,
                request.date);
```

Dependency Inversion - Example

```
17  import java.time.Month;
18  import java.time.ZoneId;
19  import java.util.Date;
20  import java.util.GregorianCalendar;
21
22  @RestController
23  public class TripCountAPI {
24
25      @Autowired
        private StationDataProvider _stationDataProvider;
27
28      public static final LocalDate launchDate = LocalDate.of(2013, Month.JUNE, 28);
29
30      @GetMapping(path="/TripsByStationPairDate", consumes="application/json")
31      ResponseEntity<Object> GetTripsByStationPairDate(
32              @Valid @RequestBody TripsByStationPairDateRequest request, Errors errors) {
33
34  //      if (errors.hasErrors()){
35  //          return ResponseEntity.status(HttpStatus.BAD_REQUEST).contentType(MediaType.APPLICATION_JSON).b
36  //      }
37  //
```

Test Driven Development

Example

```java
@RunWith(SpringRunner.class)
@WebMvcTest(TripCountAPI.class)
public class TripCountAPITest {

    @Autowired
    private MockMvc mockMvc;

    @MockBean
    private StationDataProvider sdp;
```

```java
@Test
public void tripCountAPIReturnsValid() throws Exception {

    String source = "aaaaa";
    String destination = "bbbbb";
    LocalDate date = LocalDate.now();

    Integer numTrips = 1;

    TripsByStationPairDateRequest request = makeRequest(source, destination, date);
    TripsByStationPairDateResponse response = makeResponse(request, numTrips);

    String jsonRequest = objectMapper.writeValueAsString(request);
    String jsonResponse = objectMapper.writeValueAsString(response);

    when(sdp.isValidStation(anyString())).thenReturn(value: true);
    when(sdp.getTripsByStationPairAndDate(anyString(), anyString(), any())).thenReturn(numTrips);

    mockMvc.perform(get("/TripsByStationPairDate").contentType(MediaType.APPLICATION_JSON).content(jsonReques
            .andExpect(status().isOk())
            .andExpect(content().contentType(MediaType.APPLICATION_JSON))
            .andExpect(content().json(jsonResponse));

}
```

Other Tips

Useful Tips

- Automate your testing and deployments
- Try using Docker
- Servers are cattle, not pets
- Validate any requests coming to your server
- Use frameworks - a lot of the work is already done for you!

Recommended Reading

- Clean Code - Bob Martin
- The Agile Manifesto
- The Phoenix Project - Kim, Behr, et al.
- The O’Reilly’s manual for your favorite programming language