We're Back

vervint.

For our time together:

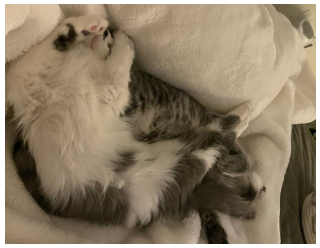
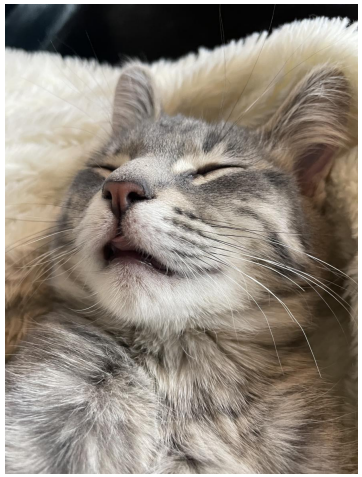
- What We've Been Up To
- What We've Learned
- A New Resource?

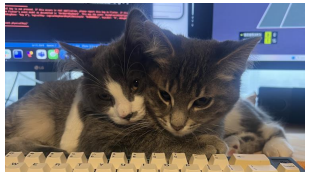
Agenda

Post Grad Life

I'm a Dad...*

Of Cats!



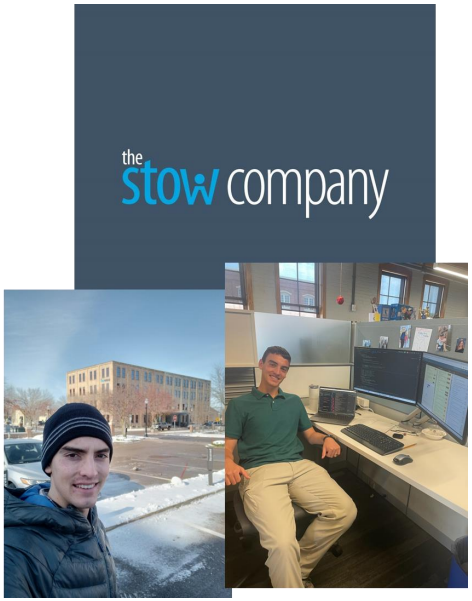
MARRIAGE AND MORE



the stow company

STARTING AT STOW

- Great experience so far
- Onboarding = drink from the firehose
  - I also got a Django tutorial from my homey Mosh
- Foosball every morning
- Learning and asking questions
- Don't be afraid to look dumb - you'll be smarter later
- Have contributed in major ways to the design tool and the easyclosets.com website redesign project

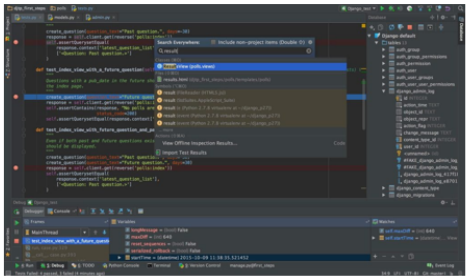


What We've Learned

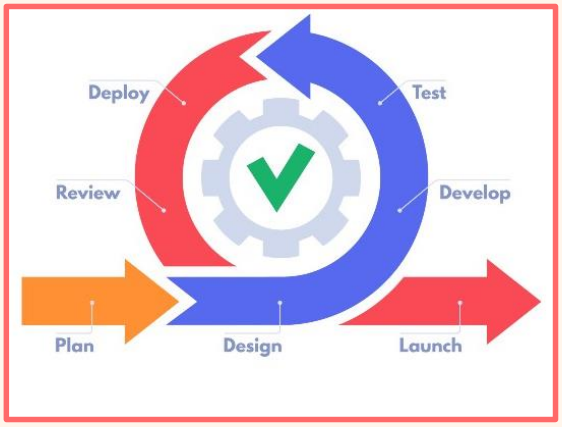
DJANGO FRAMEWORK

python™ django

Django is a high-level Python web framework designed for rapid development, clean design, and scalability. It follows the Model-View-Template (MVT) architecture and emphasizes the "Don't Repeat Yourself" (DRY) principle, allowing developers to build robust, maintainable applications efficiently. Django includes a powerful ORM for database interactions, an admin interface for content management, built-in authentication, and security features like protection against SQL injection and CSRF attacks. With its modular and extensible nature, Django is widely used for web applications ranging from simple blogs to complex, enterprise-level systems. - ChatGPT



Agile

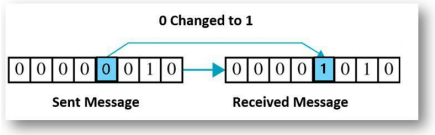


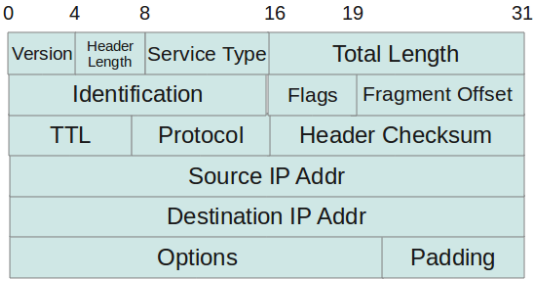
What I Love

- Honest Feedback
- Open Communication
- Devotion to Quality

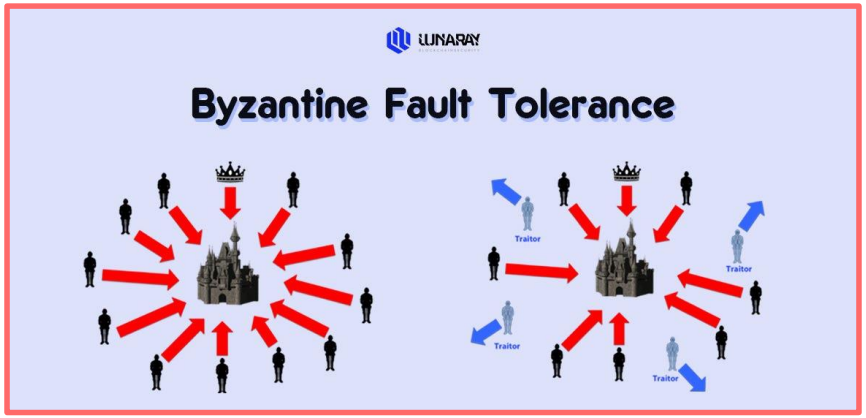
Things I wish I knew

Building Reliable Things on an Unreliable Foundation





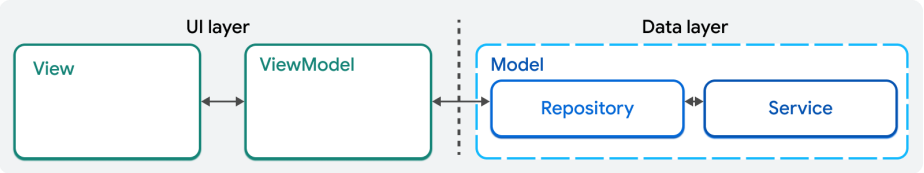
Software Problems from a Real Life POV



MVVM in Flutter

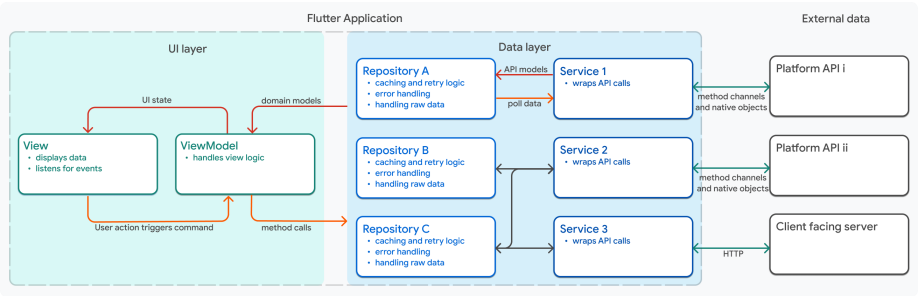
What is MVVM

Guide to app architecture

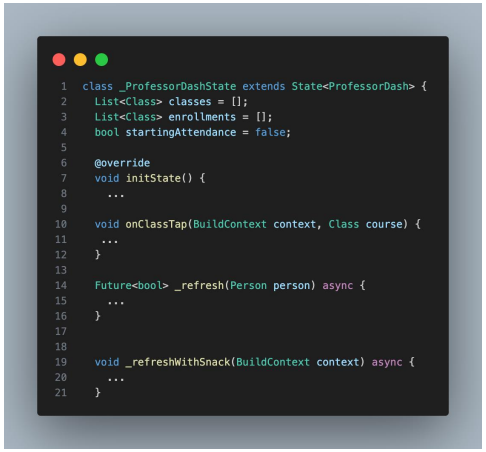


What is MvvM

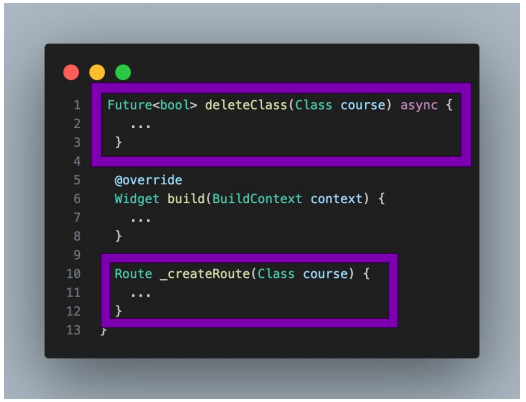
Guide to app architecture



In ImHere!



```dart
class _ProfessorDashState extends State<ProfessorDash> {
  List<Class> classes = [];
  List<Class> enrollments = [];
  bool startingAttendance = false;

  @override
  void initState() {
    ...

  void onClassTap(BuildContext context, Class course) {
    ...
  }

  Future<bool> _refresh(Person person) async {
    ...
  }


  void _refreshWithSnack(BuildContext context) async {
    ...
  }
```



```dart
Future<bool> deleteClass(Class course) async {
  ...
}

@override
Widget build(BuildContext context) {
  ...
}

Route _createRoute(Class course) {
  ...
}
```

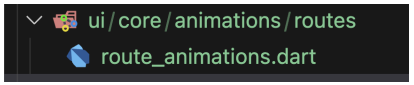
In ImHere!

```dart
Route _createRoute(Class course) {
  return PageRouteBuilder(
    pageBuilder: (context, animation, secondaryAnimation) =>
        TakingAttendance(course: course),
    transitionsBuilder: (context, animation, secondaryAnimation, child) {
      const begin = Offset(0.0, 1.0);
      const end = Offset.zero;
      const curve = Curves.ease;

      var tween = Tween(begin: begin, end: end).chain(CurveTween(curve: curve));

      return SlideTransition(
        position: animation.drive(tween),
        child: child,
      );
    },
  );
```


Code Organization

Architecture case study

```
lib
|____ui
| |____core
| | |____ui
| | | |____<shared widgets>
| | |____themes
| |____<FEATURE NAME>
| | |____view_model
| | | |_____<view_model class>.dart
| | |____widgets
| | | |____<feature name>_screen.dart
| | | |____<other widgets>
```




```dart
navigator.push(RouteAnimations.slideToTakingAttendanceRoute(course));
```

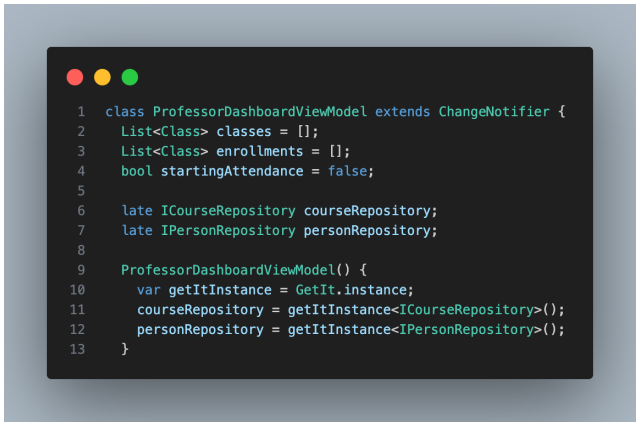
Architecture case study
https://docs.flutter.dev/app-architecture/case-study


```dart
Future<bool> deleteClass(Class course) async {
  var scaffold = ScaffoldMessenger.of(context);
  try {
    var server = GetIt.instance<ServerMethods>();
    bool response = await server.classMethods
        .deleteClass(course.id, course.professorId, course.className!);
    if (!response) {
      scaffold.showSnackBar(const SnackBar(
        content: Text("Failed to delete class"),
      ));
    }
    // remove class from list without refreshing
    setState(() {
      classes.remove(course);
    });
    return response == true; // returns false if response is null
  } catch (e) {
    scaffold.showSnackBar(const SnackBar(
      content: Text("Failed to delete class"),
    ));
    return false;
  }
}
```



```dart
Future deleteClass(BuildContext context, Class course) async {
  var scaffold = ScaffoldMessenger.of(context);

  var response = await courseRepository.deleteCourse(
      course.id, course.professorId, course.className!);

  if (!response) {
    scaffold.showSnackBar(const SnackBar(
      content: Text("Failed to delete class"),
    ));
    return;
  }

  classes.remove(course);
  notifyListeners();
}
```

```dart
abstract class ICourseRepository {
  Future<bool> deleteCourse(int courseId, int professorId, String courseName);
}

class CourseRepository implements ICourseRepository {
  final courseApi = GetIt.instance<ServerMethods>().classMethods;

  @override
  Future<bool> deleteCourse(
      int courseId, int professorId, String courseName) async {
    var success =
        await courseApi.deleteClass(courseId, professorId, courseName);

    if (!success) {
      safePrint("Failed to delete course, trying again");
      success = await courseApi.deleteClass(courseId, professorId, courseName);
    }
    return success;
  }
}

class CourseRepositoryMock implements ICourseRepository {
  @override
  Future<bool> deleteCourse(int courseId, int professorId, String courseName) {
    throw UnimplementedError();
  }
}
```

```dart
class ProfessorDashboardViewModel extends ChangeNotifier {
  List<Class> classes = [];
  List<Class> enrollments = [];
  bool startingAttendance = false;

  late ICourseRepository courseRepository;
  late IPersonRepository personRepository;

  ProfessorDashboardViewModel() {
    var getItInstance = GetIt.instance;
    courseRepository = getItInstance<ICourseRepository>();
    personRepository = getItInstance<IPersonRepository>();
  }
```

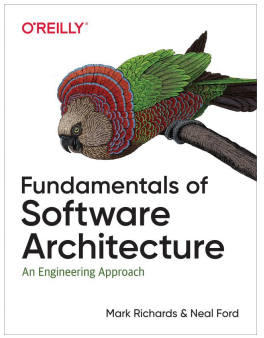

```dart
class ProfessorDash extends StatefulWidget {
  const ProfessorDash({super.key});

  @override
  State<ProfessorDash> createState() => _ProfessorDashState();
}

class _ProfessorDashState extends State<ProfessorDash> {
  List<Class> classes = [];
  List<Class> enrollments = [];
  bool startingAttendance = false;

  @override
  void initState() {
    ...

  void onClassTap(BuildContext context, Class course) {
    ...
  }

  Future<bool> _refresh(Person person) async {
    ...
  }

  void _refreshWithSnack(BuildContext context) async {
    ...
  }

  Future<bool> deleteClass(Class course) async {
    ...
  }

  void _quickTakeAttendance(BuildContext context, Class course) async {
    ...
  }

  @override
  Widget build(BuildContext context) {
    ...
  }

  Route _createRoute(Class course) {
    ...
  }
}
```

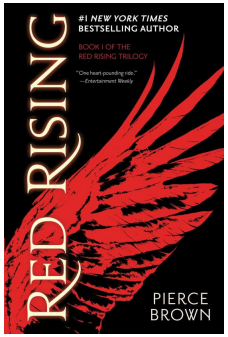

```dart
class ProfessorDash extends StatelessWidget {

  ProfessorDash({super.key});

  final ProfessorDashboardViewModel viewModel = ProfessorDashboardViewModel();

  @override
  Widget build(BuildContext context) {
    ...
  }
}
```

```dart
class ProfessorDashboardViewModel extends ChangeNotifier {
  List<Class> classes = [];
  List<Class> enrollments = [];
  bool startingAttendance = false;

  late ICourseRepository courseRepository;
  late IPersonRepository personRepository;

  ProfessorDashboardViewModel() {
    var getItInstance = GetIt.instance;
    courseRepository = getItInstance<ICourseRepository>();
    personRepository = getItInstance<IPersonRepository>();
  }

  void onTakeAttendanceTap(BuildContext context) {
    ...
  }

  void onClassTap(BuildContext context, Class course) {
    ...
  }

  Future deleteClass(BuildContext context, Class course) async {
    ...
  }

  Future<bool> refresh(BuildContext context, Person person) async {
  ...
  }

  void refreshWithSnack(BuildContext context, Person person) async {
    ...
  }

  void quickTakeAttendance(BuildContext context, Class course) async {
    ...
  }
}
```

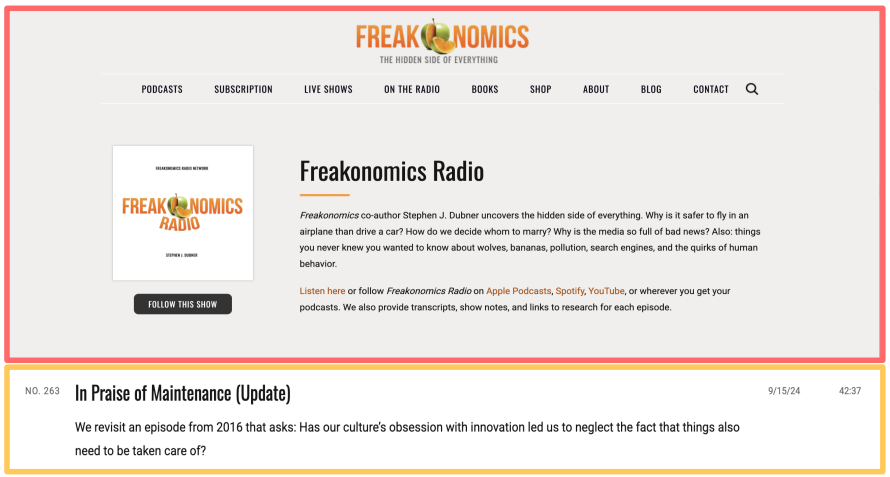
What has Helped Me

Book Recommendations





Freaknomics Podcast



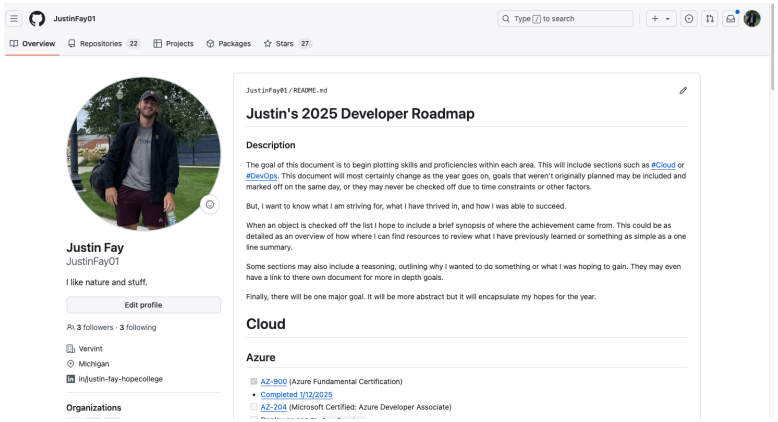
Documentation

Resource	POST	GET	PUT	DELETE
/customers	Create a new customer	Retrieve all customers	Bulk update of customers	Remove all customers
/customers/1	Error	Retrieve the details for customer 1	Update the details of customer 1 if it exists	Remove customer 1
/customers/1/orders	Create a new order for customer 1	Retrieve all orders for customer 1	Bulk update of orders for customer 1	Remove all orders for customer 1

RESTful web API design
https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design

Endless POC (Sandbox Approach)

Developer Roadmap



A New Resource

Questions — Feedback — Ideas