# Is this the Krusty Krab?

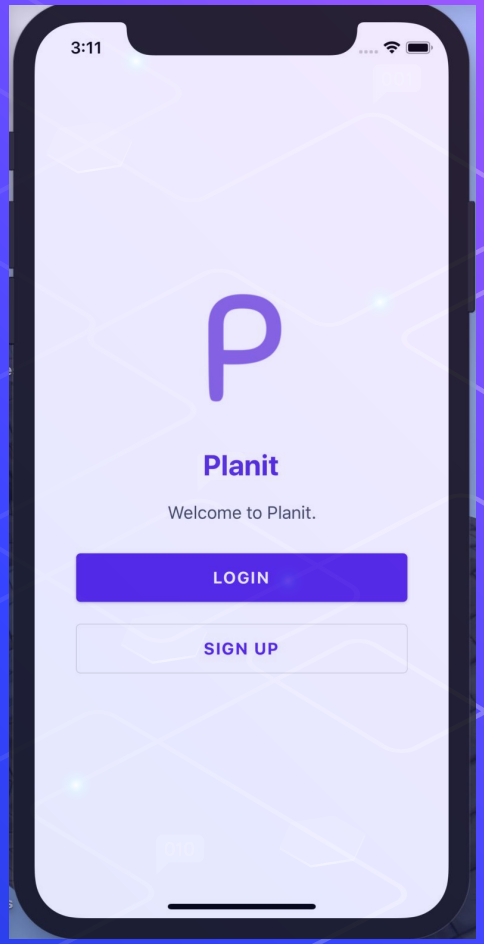
# THIS IS THE KRUSTY KRAB!

## THE TEAM

- Hemant Bhanot
- Logan Chester
- Jason Conte
- XiangQian Hong
- Jason Hu
- Gnanaswarup Srinivasan
- Jackie Tran

## OUR PURPOSE

As a team, we used scrum tactics and a REST-based MVC design to build a working implementation of Planit - a traveller-friendly mobile application envisioned by Flora Ng and Lora Haskul.



# Planit

## Key Features

1. Registration/Login
2. Itinerary Creation
3. Adding/Deleting Events
4. Google Maps API

# Feature #1: Registration/Login

- Registration and Login with data from database
- Every user has a unique itinerary
- Itineraries are saved for each user

# Feature #2: Itinerary Creation

- Itineraries are created based on filters passed by user
- Events chosen based on the filters and location
- Transportation between events chosen based on filters

# Feature #3: Adding/Deleting Events

- More events available on Explore Page
- User can add events for anytime in the itinerary
  - Event conflicts are handled
- User can delete any events in the itinerary

# Feature #4: Google Maps API

- From filters passed by user, we grab events using Google Maps API
- Gets REAL events in your proximity
- Gets REAL transportation times between events
- Show path to events on a map

# Process

# Process

## Highlights

- Scheduling Team Meetings
- Communication

## Difficulties

- Github Branching
- Learning New Technology
- API Documentation

## Techniques

- Splitting into Frontend and Backend teams
- Frontend Modular Design Pattern

# Logan Chester: Contributions

- Itinerary Creation
  - Time/Date Filters
  - Budget/Distance Filters
  - Location Selection
- Itinerary Event Mapping
  - Front-End Map Routing Implementation

# XiangQian Hong: Contributions

- Rendering of Itinerary
  - Events
  - Transportation
- Front-End Architecture
- Add/Delete Events

# Gnanaswarup Srinivasan: Contributions

- User Registration/Login
  - App entry screen
  - Login/register screens
  - Navigation logic between the screens
  - RegEx pattern matching for email validity
- User Location Selection

# Hemant Bhanot: Contributions

- Google Maps Places API
  - Nearby place search queries/requests
    - Max Distance
    - Types of activities
    - Budget (Price level)
- Google Maps Routes API
  - Transportation between events
    - Types of transportation
    - Cost and location

# Jason Conte: Contributions

- Itinerary
  - Creation Algorithm
  - Event Scoring Algorithm
- Users
  - Objects
  - Database
- Event Database

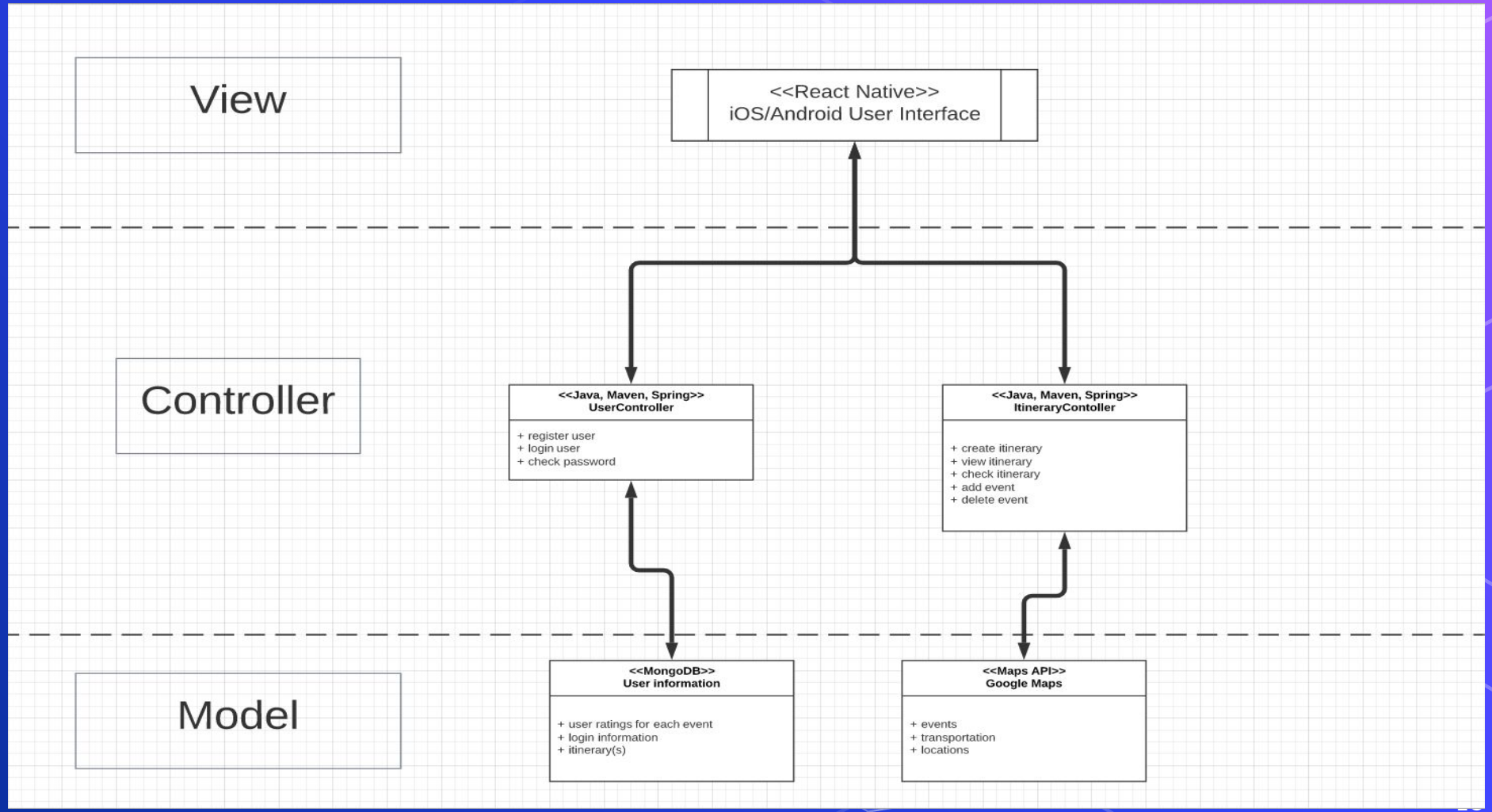
# Jason Hu: Contributions

- Itinerary Manipulation
  - Adding Events
  - Deleting Events
  - Event Time Evaluation filter
- REST Endpoint Mappings for storing and providing itinerary information
- Software architecture diagram w/ Jackie

# Jackie Tran: Contributions

- Itinerary Manipulation
  - Adding Events (Handling Conflicts)
- Explore Page Data
  - Choosing Events to Send to FrontEnd
- Useful REST endpoints for Frontend Team

# Software Architecture


View
<<React Native>>
iOS/Android User Interface
Controller
<<Java, Maven, Spring>>
UserController
+ register user
+ login user
+ check password
<<Java, Maven, Spring>>
ItineraryContoller
+ create itinerary
+ view itinerary
+ check itinerary
+ add event
+ delete event
Model
<<MongoDB>>
User information
+ user ratings for each event
+ login information
+ itinerary(s)
<<Maps API>>
Google Maps
+ events
+ transportation
+ locations

# Model

## MongoDB

- Store users information and itinerary(s)

## Google Maps API

- Called to obtain:
  - Events
  - Transportation

# Itinerary Controller

- Create Itinerary
- View Itinerary
- Add Events
- Delete Events

```java
@PutMapping("/deleteEvent")
public ResponseEntity<?> deleteEvent(@RequestBody Map<String, String> body) {
  String eventId = new String(body.get("eventId"));
  user.getItinerary().deleteEvent(eventId);
  mpd.updateUser(user);
  return ResponseEntity.ok().build();
}

@PutMapping("/addRating")
public ResponseEntity<?> addRating(@RequestBody Map<String, String> body) {
  user.getEventRatings().put(body.get("eventId"), Integer.parseInt(body.get("rating")));
  mpd.updateUser(user);
  return ResponseEntity.ok().build();
}

@PutMapping("/deleteRating")
public ResponseEntity<?> deleteRating(@RequestBody Map<String, String> body) {
  user.getEventRatings().remove(body.get("eventId"));
  mpd.updateUser(user);
  return ResponseEntity.ok().build();
}

@PutMapping("/addVisitedEvent")
public ResponseEntity<?> addVisitedEvent(@RequestBody Map<String, String> body) {
  user.getVisitedEvents().add(body.get("eventId"));
  mpd.updateUser(user);
  return ResponseEntity.ok().build();
}
```

# User Controller

- User Registration
- User Login
- User Credentials
- Fetch Itinerary

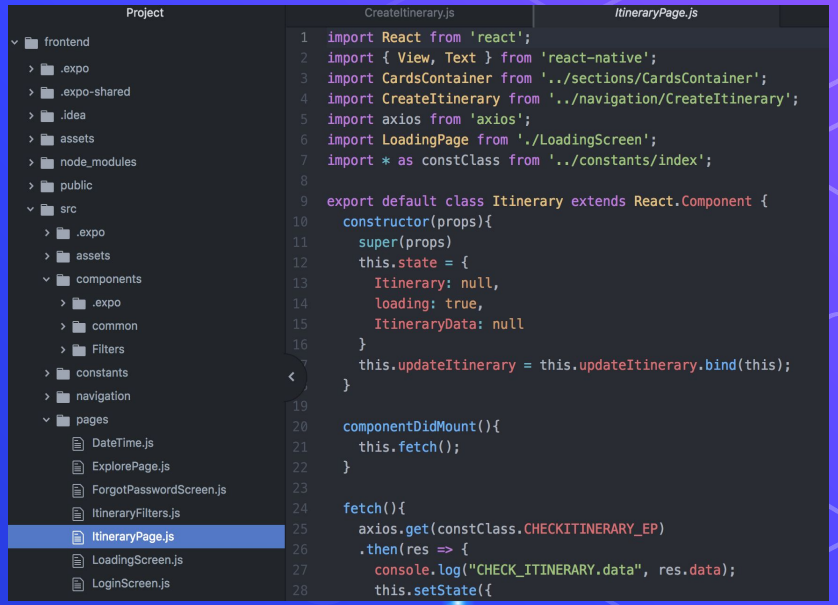
```java
@PutMapping("/login")
public void login(@RequestBody User body) { this.user = mpd.readUser(body.getEmail()); }

@PostMapping("/checkPw")
public String checkPw(@RequestBody Map<String, String> body) {
  User user = mpd.readUser(body.get("email"));
  if(user != null && user.getPassword().equals(body.get("password"))){
    return "valid";
  }
  else{
    return "invalid";
  }
}

@PostMapping("/register")
public void register(@RequestBody Map<String, String> body) {
  System.out.println(body);
  this.user = new User();
  this.itin = this.user.getItinerary();
  this.user.setUsername(body.get("username"));
  this.user.setPassword(body.get("password"));
  this.user.setEmail(body.get("email"));
  mpd.createUser(this.user);
  // userService.addUser(user);
}
```

# View (Client)

- React Native
- Modular Design Pattern
- Handles User Data Input
- Interacts with Controllers to Render Application using REST Endpoints



# Tools

- Java
- MongoDB
- Google Maps API
- React Native
- SpringBoot
- Maven
- Jira
- Postman
- Git/Github
- Expo

# Technical Challenges

- React Native State Management
- React Native Dynamic Photo Rendering
- Google Maps API Documentation Outdated
- Cost of API Calls (Testing and Demo)

# Thank You For Listening!

Please feel free to ask questions.