Group 10
John Gilbert
Mark Howell
Jason Mathew
Courtney Stewart
Joseph Tolbert

orlando math circle
OMC

Sponsors
Margarita Azbel
Sheina Rodriguez
Matthew Villegas

Introductions

- John Gilbert — Project Manager, Lead Backend Developer
- Mark Howell — Lead Frontend Designer
- Jason Mathew — Frontend Designer
- Courtney Stewart — State Management & Backend Developer
- Joseph Tolbert — Backend Developer

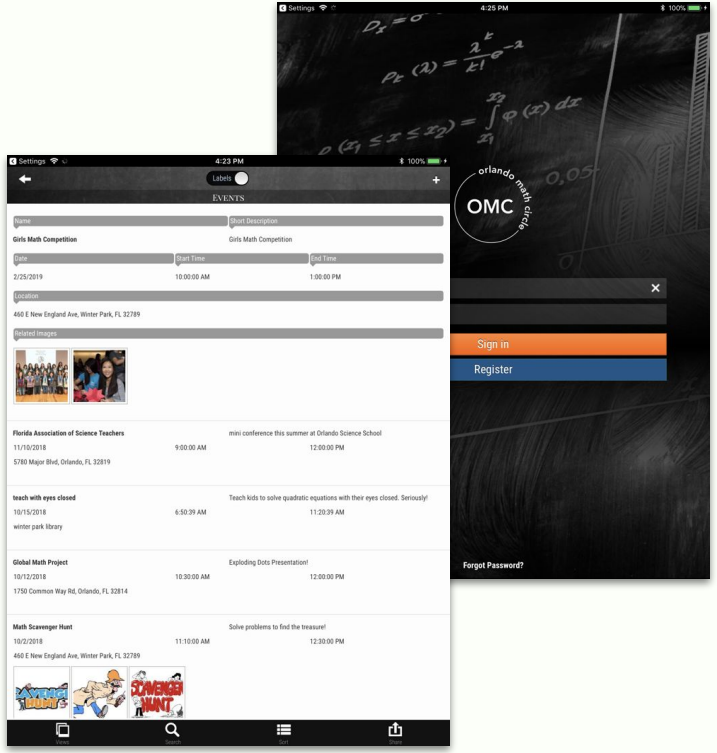
Who is Orlando Math Circle (OMC)?

Orlando Math Circle aims to demystify mathematics for students of all ages through playful, joyous problem solving

- **Mission** — Create a diverse and inclusive community of student mathematicians
- **Strategy** — Provide engaging and fun opportunities to do mathematics outside of school
- **Vision** — Equitable access and greater student participation in mathematics
- 501(C)(3) non-profit organization

Existing Solution

- Closed-source mobile app
- 7-step registration process that requires desktop
- Unintuitive interface
- No integration for tracking volunteers



Project Goals & Requirements

We are developing an event calendar system with an app-like experience on a mobile browser.

- Event management and check-in system
- PayPal payments for events with fees
- Volunteer management
- Email notifications for events and newsletters
- Extended usability through admin panel
- Open source, documented, and extensible

Frontend

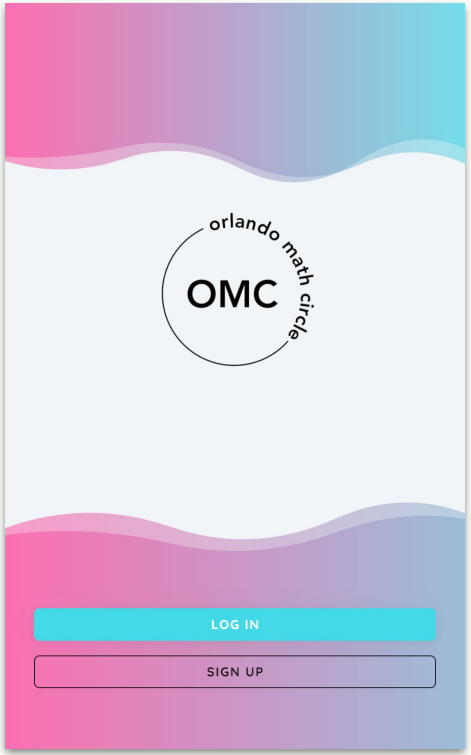
Frontend Overview

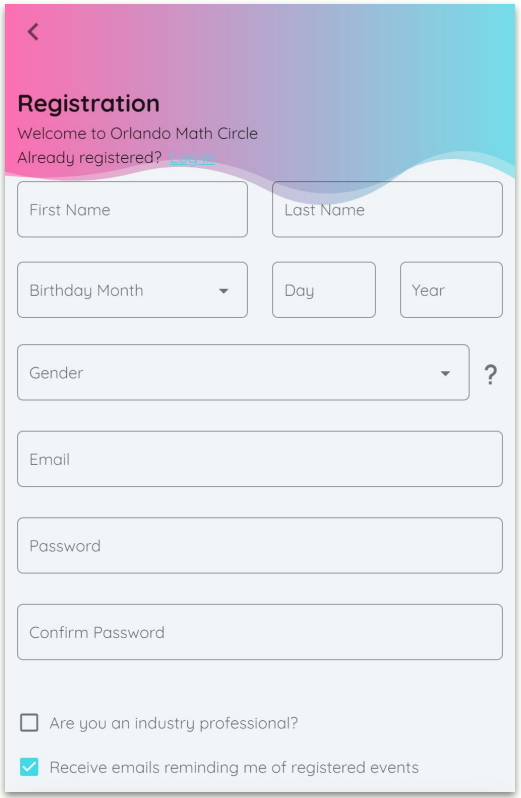
- Vue.js through the Nuxt.js framework
- Server-Side Rendered (SSR)
  Single-Page Application (SPA)
- Vuetify component library
- Separation of concerns in single-file components
- Assets and design direction by Sheina

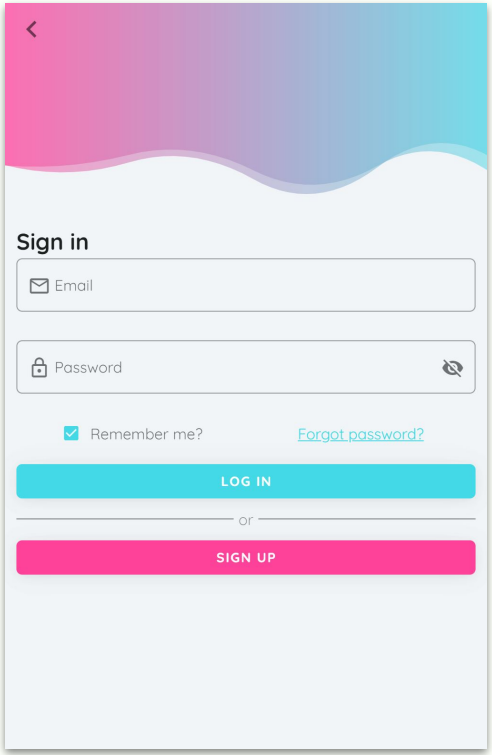
NUXTJS

```
<template>
  <div>
    <v-toolbar flat color="transparent">
      <v-btn icon to="/">
        <v-icon large>mdi-chevron-left</v-icon>
      </v-btn>
    </v-toolbar>

    <v-container>
          ...
    </v-container>
  </div>
</template>

<script>
export default {
  layout: 'landing',
  auth: 'guest',
}
</script>

<style lang="scss" scoped>
.example {
  a {
    text-decoration: none;
  }
}
</style>
```

Landing Pages

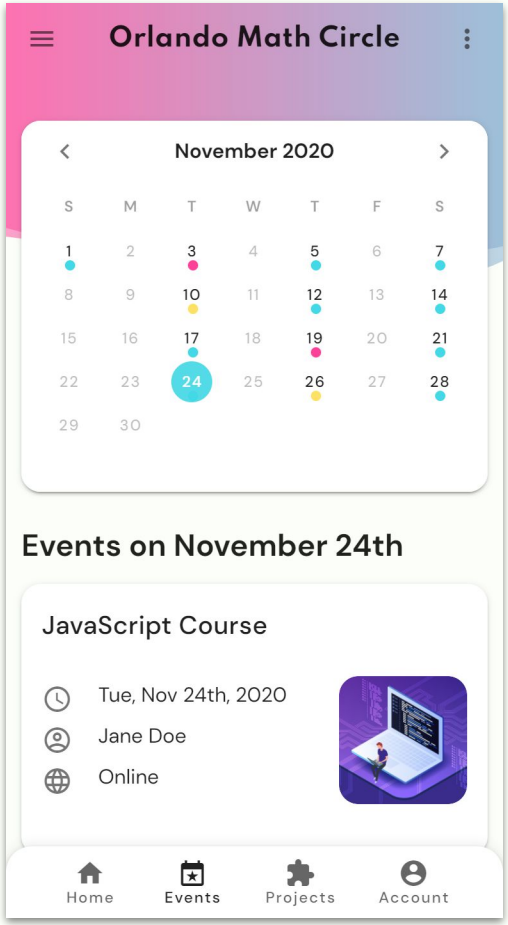


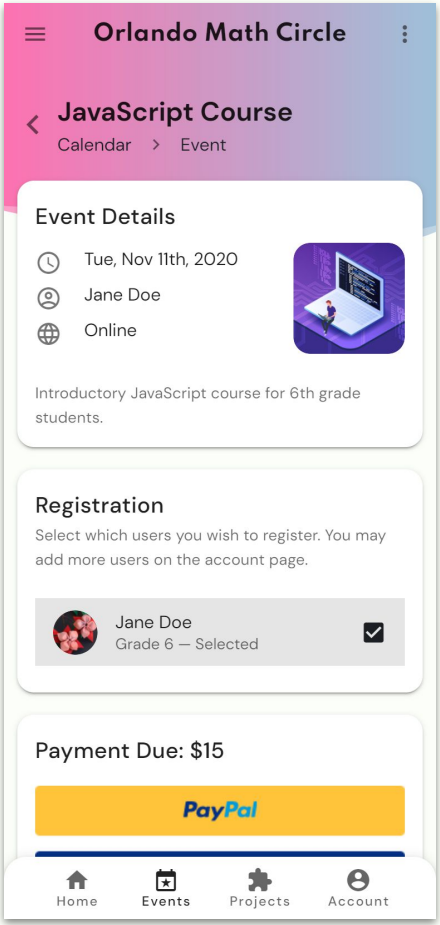




Events Page

- Configurable calendar of events
- Sub-event page for registering to events
- Adults can register multiple users at once
- Fee payment through PayPal Orders API
- Users can register as volunteers and pick a job



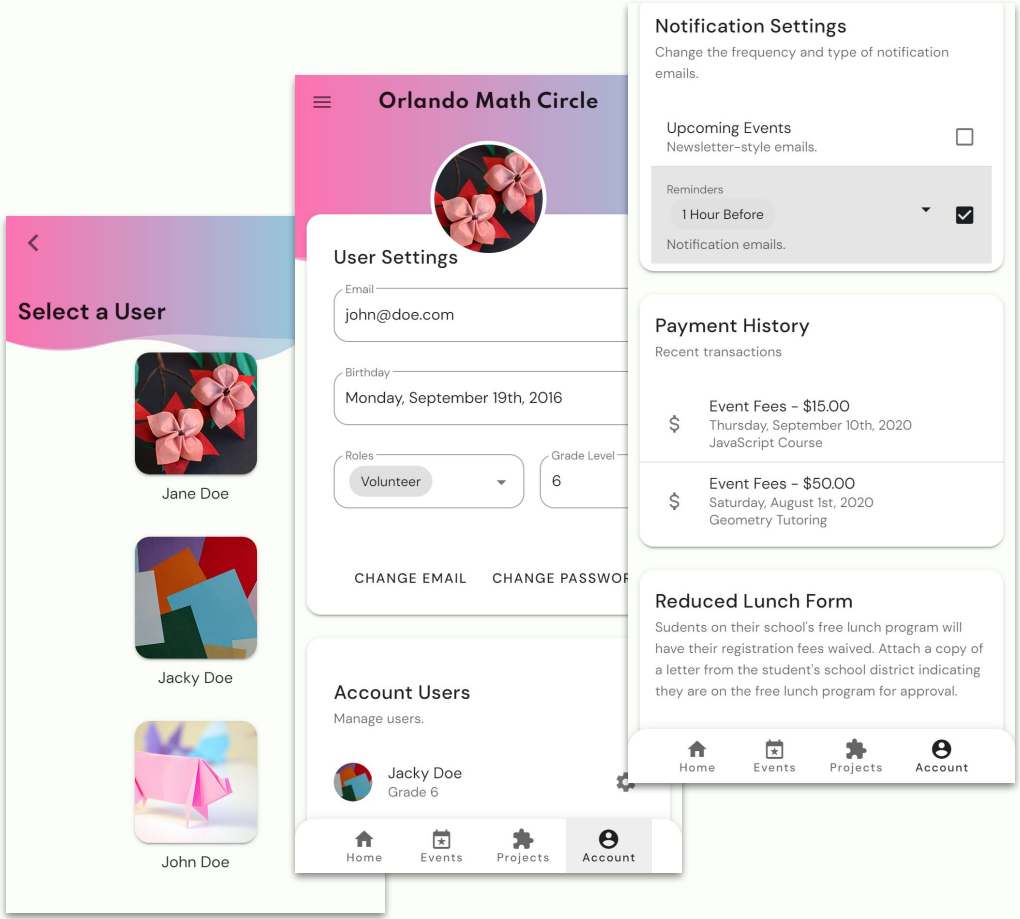


Projects Page

- Versatile page with function based on the user's role
- Discovery tool for users to learn about the types of events hosted by OMC
- Projects are groupings of related events
- Courses are blocks of project events

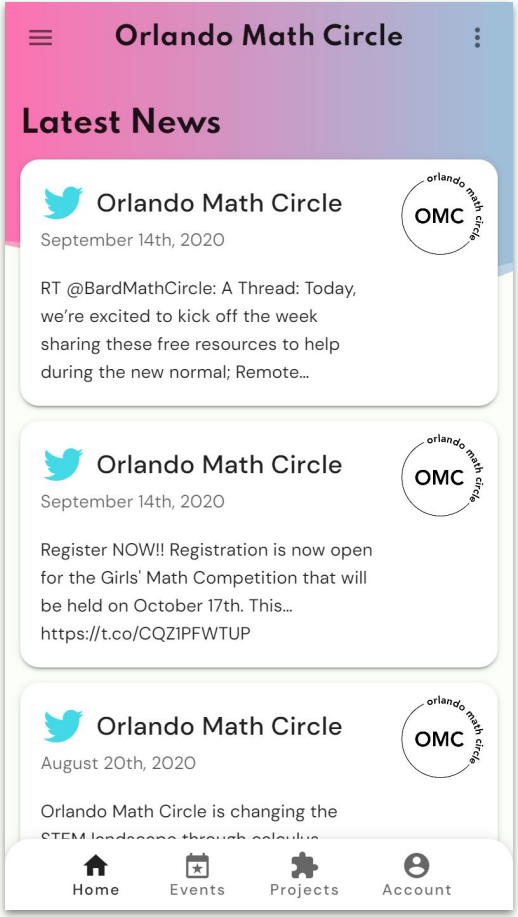
Account Page & Switcher

- Customizable email notifications
- Volunteer and reduced lunch forms
- Adults can add other users to an account
- Switch easily between any user from the navigation bar
- Transactional user actions
- View payment history



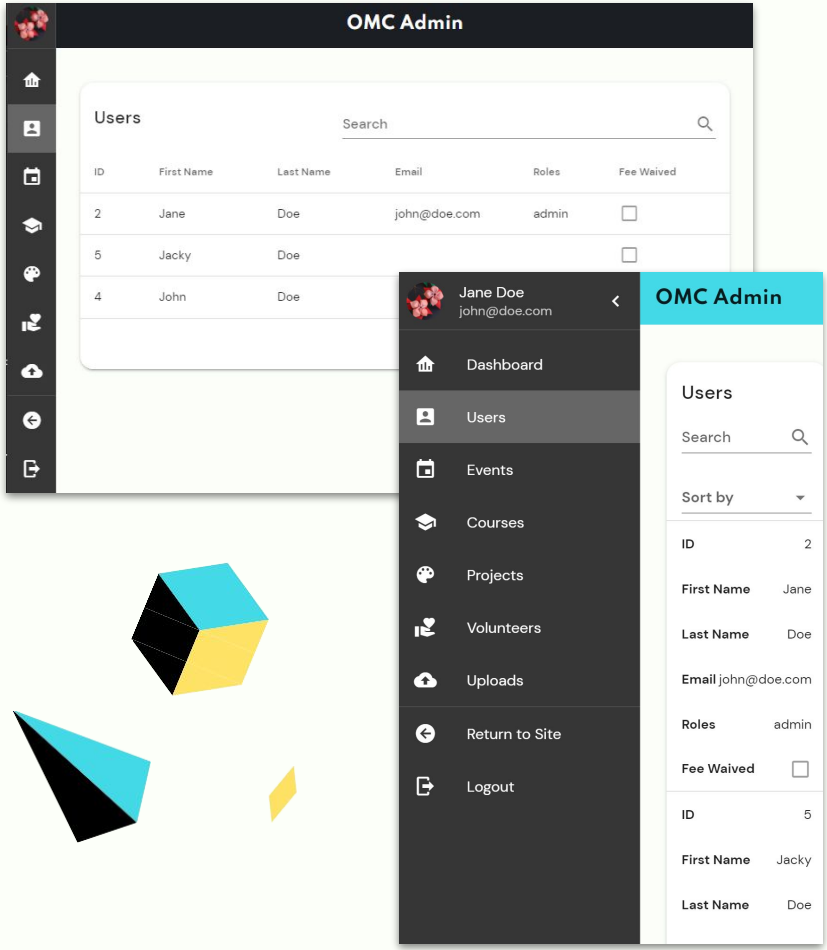
Home Page

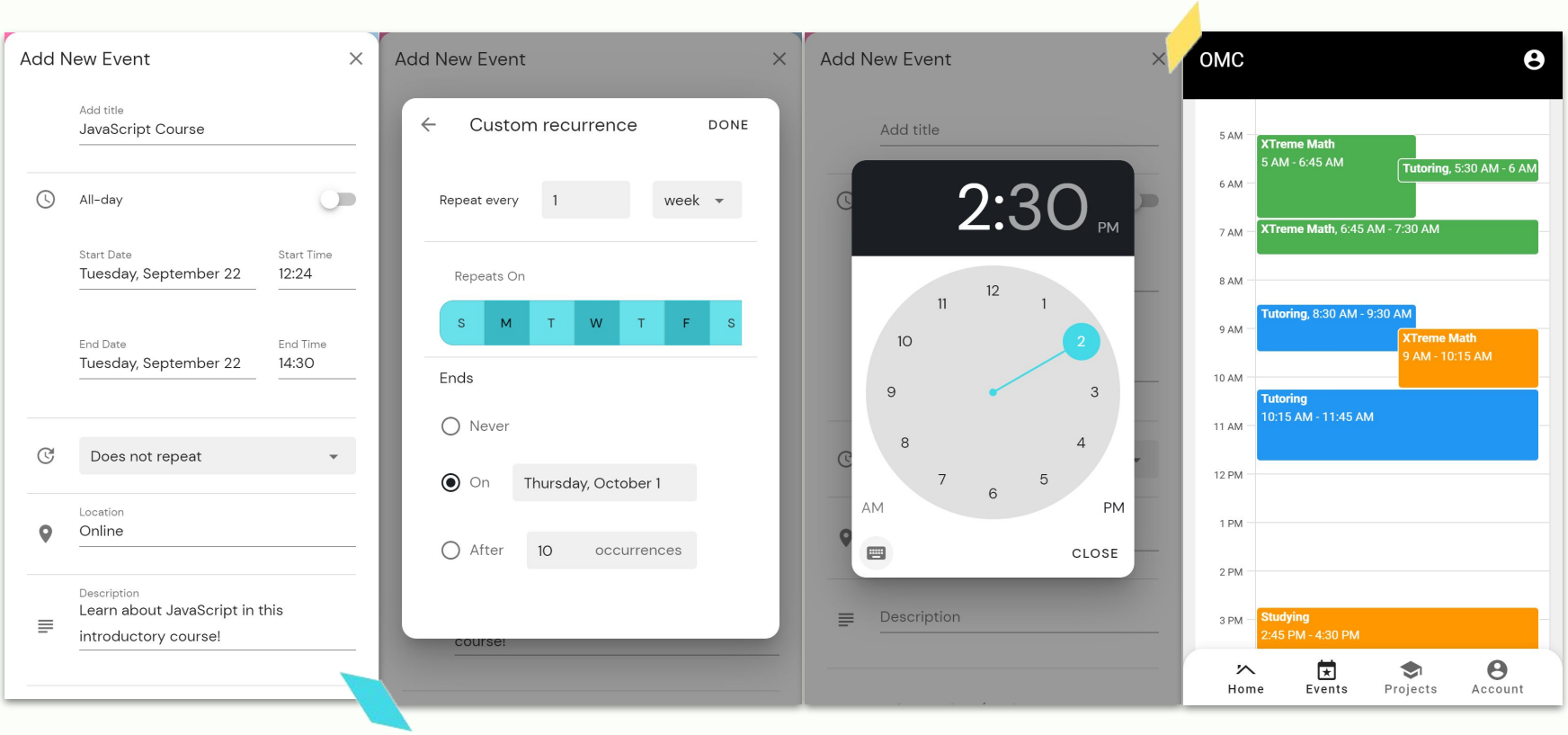
- Landing page for news and upcoming events
- Twitter integrated news feed
- Notification bar for important user information



Admin Panel

- Available only to users with the administrator role
- Create, update, or destroy any entity
- Search users by grade level, genders, ages, or roles
- Email users en-masse through the panel or by retrieving the emails comma-separated
- Dashboard with basic statistics
- Home of the captive attendance page
- Tabular data best suited for larger devices, but will operate just the same on mobile



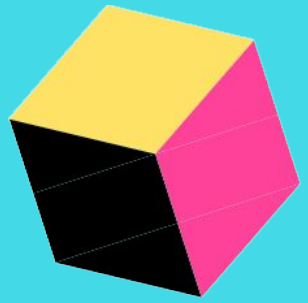

Add New Event
Add title
JavaScript Course
All-day
Start Date
Tuesday, September 22
Start Time
12:24
End Date
Tuesday, September 22
End Time
14:30
Does not repeat
Location
Online
Description
Learn about JavaScript in this introductory course!
Add New Event
Custom recurrence
DONE
Repeat every
1
week
Repeats On
S M T W T F S
Ends
Never
On
Thursday, October 1
After
10
occurrences
Add New Event
Add title
2:30 PM
AM
PM
CLOSE
Description
OMC
5 AM
6 AM
7 AM
8 AM
9 AM
10 AM
11 AM
12 PM
1 PM
2 PM
3 PM
XTreme Math
5 AM - 6:45 AM
Tutoring, 5:30 AM - 6 AM
XTreme Math, 6:45 AM - 7:30 AM
Tutoring, 8:30 AM - 9:30 AM
XTreme Math
9 AM - 10:15 AM
Tutoring
10:15 AM - 11:45 AM
Studying
2:45 PM - 4:30 PM
Home
Events
Projects
Account

Vuex State Management

- Vue counterpart to React's Redux
- Event data is shared across homepage, projects, and events pages in a global state
- Components utilizing Vuex state react to changes automatically
- Retrieves auth and settings cookies on first load
- Attaches JWT as bearer token in each request
- Can reuse TypeScript definitions from backend for easier implementation

```
export const state = () => ({
  token: {
    jwt: null as string | null,
    complete: false,
    remember: true,
  },
  user: null as User | null,
})

export type AuthState = ReturnType<typeof state>

export const getters: GetterTree<AuthState, AuthState> = {
  loggedIn: (state) => state.token.jwt && state.token.complete,
  isAdmin: (state) => state.token.jwt && state.token.complete &&
    state.user?.roles?.includes(Roles.ADMIN),
}

export const mutations: MutationTree<AuthState> = {
  SET_STATUS: (state, status: StateStatus) => (state.status = status),
  SET_TOKEN: (state, data: AuthState['token']) =>
    (state.token = Object.assign({}, state.token, data)),
  SET_USER: (state, user: User) => (state.user = user),
}

export const actions: ActionTree<AuthState, AuthState> = {
  async login({ commit, dispatch }, { email, password, remember }) {
    commit('SET_STATUS', StateStatus.BUSY)

    const { token, complete } = await this.$axios.$post('/login', {
      email,
      password,
    })

    dispatch('setTokenCookie', { jwt: token, complete, remember })

    commit('SET_STATUS', StateStatus.WAITING)
  },
}
```

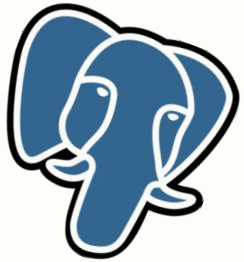
Backend



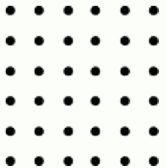
Backend Overview

- PostgreSQL database with MikroORM
- Node.js and the Nest.js (Express Framework)
- Modular, class-based, and opinionated
- TypeScript and custom third-party interfaces
- Automatic crash recovery with PM2, a NPM library for app process management and restarts

Why PostgreSQL?

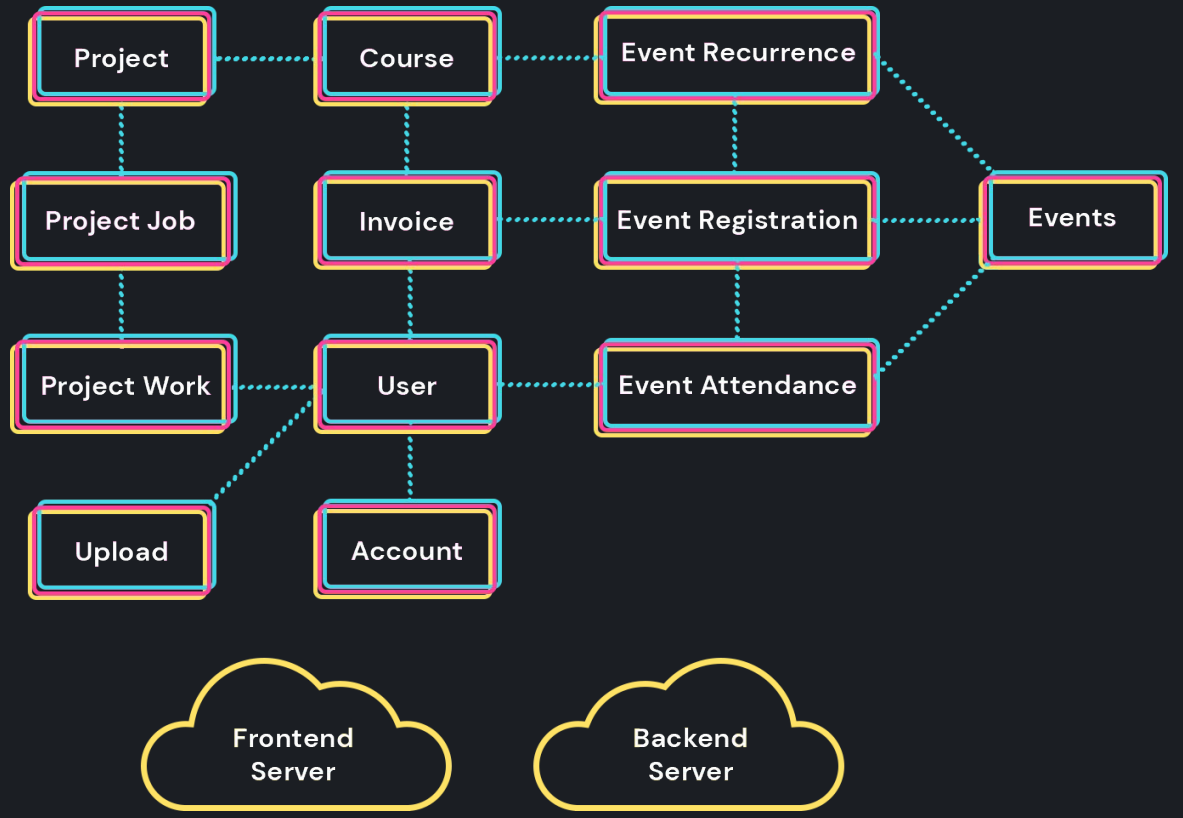


- Desire for joining data and reverse lookups eliminated NoSQL
- Constraints help enforce COPPA compliance and referential integrity
- PostgreSQL has a faster development schedule compared to MySQL
- JSON storage with query and index support through the *jsonb* type
- Automatic schema generation and TypeScript support with MikroORM



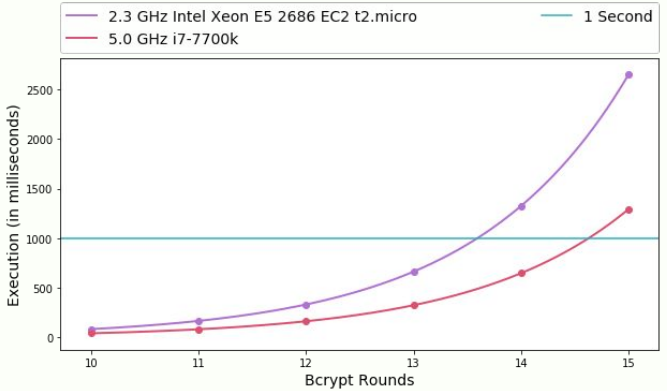
Entity Diagram

- Entities are generalized wherever possible for future development
- Constraints mapped to HTTP status errors for better developer experience (DX)



Accounts & Users

- Accounts have a primary user who owns the login credentials
- Semi-authenticated state after initial login which requires user selection for a full token
- Stateful JWTs stored as cookie
- Password hashed and salted with bcrypt
- Custom RBAC permissions decorators




```
@AccountAuth()
@Get('me')
getMe(@Acc() account: Account) {
  return account;
}

@UserAuth('account', 'read:any')
@Get(':id')
findOne(@Param() { id }: FindOneAccountDto) {
  return this.accountService.findOneOrFail(id);
}
```

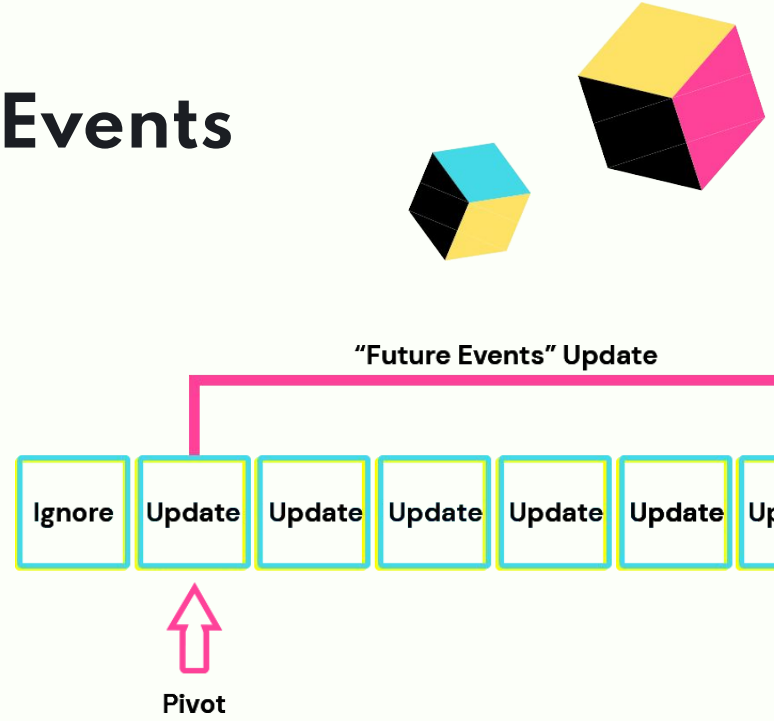
Events & Recurrence

- Events are metadata associated with a UTC **start** and **end** date
- Recurring events are described by the iCalendar RFC5545 **RRule** specification
- RRule created, serialized, and deserialized through the **rrule.js** library
- On-demand hydration of events
- Infinitely recurring events possible, but not desired

```
// Every week until January 30, 2021
const rrule = new RRule({
  freq: Frequency.WEEKLY,
  dtstart: new Date(Date.UTC(2020, 11, 1, 0, 0, 0)),
  until: new Date(Date.UTC(2021, 0, 30, 0, 0, 0)),
});

// Serialize (store in database)
rrule.toString();

// DTSTART:20201201T000000Z
// RRULE:FREQ=WEEKLY;UNTIL=20210130T000000Z;INTERVAL=1;WKST=MO

// Deserialize (get from database)
const recreatedRRule = rrulestr(...string);
```

Modifying Recurring Events

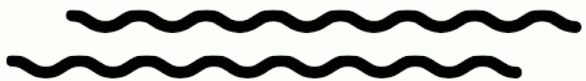
- Modifications can occur to individual events, future events, or to all events
- Deleting individual events stores an exception in the rrule
- Metadata changes are non-destructive
- Store original time if a single event is updated to avoid re-creating it later
- *All* or *Future* updates select a pivot, update it, then check if the other events can be updated or deleted

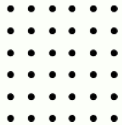


Event Registration

- Registration process
  a. Receives an array of user ids and the event id
  b. Compare user attributes to permissions structure on the event and validate
  c. If the event has a course, retrieve it to determine payment modality
  d. Ensure for each user there is a paid invoice
  e. Generate registration records for each user
- Volunteers bypass normal registration requirements
- Registrations contain volunteering job and work information

```
const capturedOrder = await this.paypalService.captureOrder(orderId);
const createInvoiceDtos: CreateInvoiceDto[] = [];

for (const purchase_unit of capturedOrder.purchase_units) {
  const capture = purchase_unit.payments.captures[0];

  createInvoiceDtos.push({
    status: InvoiceStatus.COMPLETED,
    id: capture.id,
    fee: capture.seller_receivable_breakdown.paypal_fee.value,
    gross: capture.seller_receivable_breakdown.gross_amount.value,
    net: capture.seller_receivable_breakdown.net_amount.value,
    purchasedAt: new Date(capture.create_time),
    event,
    user: +purchase_unit.reference_id,
  });
}

return this.invoiceService.batchCreate(createInvoiceDtos);
```

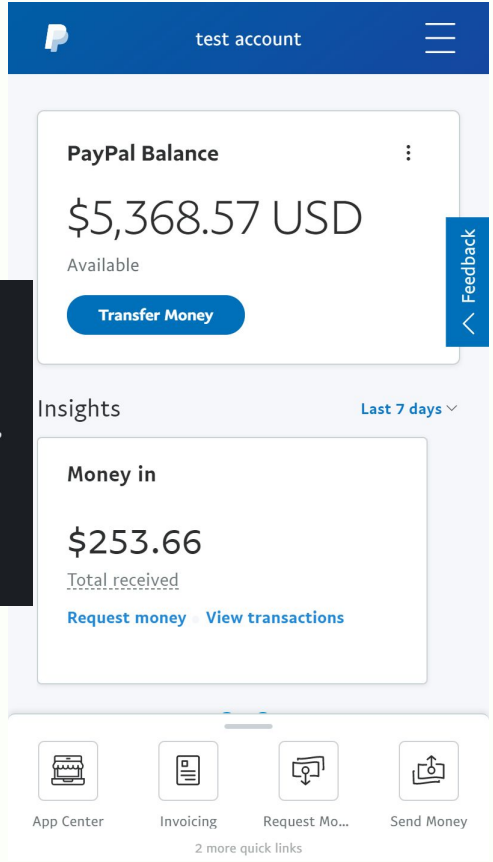




PayPal and Invoices

- PayPal Orders API utilizes OAuth 2.0
- Custom token service creates a bearer token, replaces expiring tokens, and retries requests on authentication errors
- Orders API only cares about cost (in $)
- PayPal Smart Buttons flow:
    a. Server **creates** an order, 1 unit per user.
    b. User confirms order and popup closes
    c. Server **captures** the order, then creates the invoice by mapping items to users
- No direct refunds are made to the user

```
{
    "purchase_units": [
        {
            "amount": {
                "currency_code": "USD",
                "value": "15.00"
            }
        }
    ]
}
```



Event Attendance

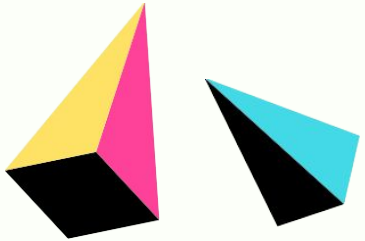
- Attendance page maintains an updated list of users and registrations through websockets
- Nest.js provides Socket.IO integration through a gateway interface
- Gateways accept subscribe messages used to join a room for the event
- Room is injected into account creation and registration methods to push changes
- Secured through WSS (SSL/TSL) and modified authentication flow

```
@WebSocketGateway(80, { namespace: 'events' })
export class EventAttendanceGateway {
  @SubscribeMessage('join')
  onJoin(
    @MessageBody() { eventId }: JoinEventRoomDto,
    @ConnectedSocket() client: Socket,
  ) {
    client.join(`event-${eventId}`)
  }
}

// Elsewhere...A user was just registered

@WebSocketServer()
server: Server;

this.server.emit(`event-${registration.event.id}`, registration)
```

Courses & Projects

- Courses are a unique group of events that describe payment modality
- Projects are reusable metadata for events and courses to describe types of events
- Retrieved for the projects page to display metadata, or get payment requirements

Project	Programming Courses
Course	Intro to JavaScript
Events	Meeting Times



Volunteering

- Volunteers pick jobs during event registration
- Separate permissions on events for volunteers
- Jobs are associated with projects or events
- Project work is recorded during attendance
- Assumes work hours are completed, but can be deducted by admins in the panel
- Volunteer role is added through a volunteer handbook form in the account panel
- Generalized work is created through an approval process on a separate form

```
async create(eventId: number, user: User, jobId: number) {
  const event = await this.eventService.findOneOrFail({
    id: eventId,
    jobs: {
      id: jobId
    }
  }, ['jobs']);

  // Check for existing registrations, invoices, and permissions

  const work = this.projectWorkService.create(user, jobId);

  user.work.push(work);

  const registration = this.registrationRepository.create({ user, event })

  await this.registrationRepository.persist(registration).flush();

  return registration;
}
```

Emailing & Notifications

- SendGrid API through Azure
- Transactional and newsletter templates with dynamic variable substitutions
- Nest.js module for cron-style scheduling
- Runs scan to send new notifications every 15 minutes
- Stores triggered notifications to avoid sending emails repeatedly

```
@Injectable()
export class EmailScheduler extends NestSchedule {
  constructor(@InjectQueue('email') private readonly queue: Queue) {
    super();
  }

  // Every 15th minute
  @Cron('*/15 * * * *')
  enqueueNotifications() {
    this.queue.add('send-email-notifications', { attempts: 1 });
  }
}

@Processor('email')
export class EmailQueue {
  @Process({ name: 'send-email-notifications', concurrency: 1 })
  private async sendEmailNotifications(job: Job) {
    // Find event registrations and send email notifications
  }
}
```

File Uploads

- Files are uploaded using the *multipart/form-data* content type
- Nest.js provides a *Multer* library interceptor that extracts files to the disk
- File size and MIME type enforcement
- Files served statically for admin download
- File names hashed and not indexed for security

```
@Post('upload')
@UseInterceptors(FilesInterceptor('form', 1))
uploadForm(@Usr() user: User, @UploadedFiles() files) {
  return this.uploadService.create(user, files)
}
```

Networking

- NGINX reverse proxy server
- Intercepts requests and forwards them to the frontend or backend
- Caches and serves static assets
- Contains *Let's Encrypt* SSL certificate credentials and renewal flow
- Frontend routed to main domain
- Backend to subdomain *api.orlandomathcircle.org*

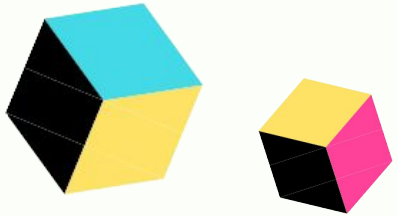
```
map $sent_http_content_type $expires {
    "text/html"                 epoch;
    "text/html; charset=utf-8"  epoch;
    default                     off;
}

server {
    listen          80;
    listen          443;
    server_name     example.com;

    gzip            on;
    gzip_types      text/plain application/xml text/css application/javascript;
    gzip_min_length 1000;

    location / {
        expires $expires;

        proxy_redirect                    off;
        proxy_set_header Host             $host;
        proxy_set_header X-Real-IP        $remote_addr;
        proxy_set_header X-Forwarded-For  $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_read_timeout          1m;
        proxy_connect_timeout       1m;
        proxy_pass                        http://127.0.0.1:8000; # Frontend port
    }
}
```

Budget

- Non-Profit $3500 Yearly Credit with Microsoft Azure
- Simple dashboard for incorporating Azure services
- Provides SendGrid emailing with 25,000 emails a month for free
- Provides Office 365 to Orlando Math Circle for free
- No external costs to OMC

VM Instance	vCPU	RAM	Transfer	Storage	Cost
BS2	2 vCPUs	4 GB	$0.02/GB	$0.30/GB	$36.21/Mo

Stage 2



- Core of application is nearly feature-complete
- Further development and integrations can simplify OMC operations
- May require maintenance and modification once adopted
- Account balance, quizzing, points system, student sandbox and per-event custom behaviors were discussed features that could be revisited
- These additional features would distinguish OMC from other math centers

Questions?