Dashboard.html

<!-- src/app/components/dashboard/dashboard.component.html -->

<div class="dashboard-container">

    <div class="sidebar">

      <div class="logo-container">

        <img src="assets/images/equihub-logo.png" alt="EquiHub" class="logo" />

        <span class="logo-text">EquiHub</span>

      </div>

      <div class="nav-links">

        <a

          [routerLink]="['/dashboard/rsu']"

          routerLinkActive="active"

          [ngClass]="{'active-rsu': activeTab === 'rsu'}"

          (click)="activeTab = 'rsu'"

          class="nav-link"

        >

          RSU

        </a>

        <a

          [routerLink]="['/dashboard/esop']"

          routerLinkActive="active"

          [ngClass]="{'active-esop': activeTab === 'esop'}"

          (click)="activeTab = 'esop'"

          class="nav-link"

        >

          ESOP

        </a>

        <a

          [routerLink]="['/dashboard/vest']"

          routerLinkActive="active"

          [ngClass]="{'active-vest': activeTab === 'vest'}"

          (click)="activeTab = 'vest'"

          class="nav-link"

        >

          Vest

        </a>

      </div>

      <button class="logout-btn" (click)="logout()">Logout</button>

    </div>

    <div class="content-container">

      <router-outlet></router-outlet>

      <div class="footer">

        <div class="footer-section">

          <h4>EquiHub</h4>

          <a [routerLink]="['/dashboard/about']">About</a>

          <a [routerLink]="['/dashboard/features']">Key Features</a>

          <a [routerLink]="['/dashboard/news']">News & Updates</a>

          <a>Equity Overview</a>

        </div>

        <div class="footer-section">

          <h4>Help & Reference</h4>

          <a [routerLink]="['/dashboard/faq']">FAQs</a>

          <a [routerLink]="['/dashboard/glossary']">Glossary of Terms</a>

        </div>

        <div class="footer-section">

          <h4>Important Information</h4>

          <a [routerLink]="['/dashboard/tax-info']">Tax Information</a>

          <a [routerLink]="['/dashboard/legal']">Legal Notices</a>

        </div>

        <div class="footer-section">

          <h4>Connect & Contribute</h4>

          <a [routerLink]="['/dashboard/feedback']">Suggestions & Feedback</a>

          <a>Accessibility Feedback</a>

          <a [routerLink]="['/dashboard/support']">Contact Support</a>

        </div>

        <div class="social-links">

          <a href="#" class="social-icon">

            <i class="fa fa-facebook"></i>

          </a>

          <a href="#" class="social-icon">

            <i class="fa fa-instagram"></i>

          </a>

          <a href="#" class="social-icon">

            <i class="fa fa-envelope"></i>

          </a>

          <a href="#" class="social-icon">

            <i class="fa fa-twitter"></i>

          </a>

          <a href="#" class="social-icon">

            <i class="fa fa-linkedin"></i>

          </a>

        </div>

      </div>

      <div class="copyright">

        Copyright© 2025 EquiHub. All Rights Reserved.

      </div>

    </div>

  </div>

Dashboard.ts

// src/app/components/esop-page/esop-page.component.ts

import { Component, OnInit } from '@angular/core';

import { CommonModule } from '@angular/common';

import { AuthService } from '../../services/auth.service';

import { CalculationService } from '../../services/calculation.service';

import { User, Symbol } from '../../models/user';

@Component({

  selector: 'app-esop-page',

  standalone: true,

  imports: [CommonModule],

  templateUrl: './esop-page.component.html',

  styleUrls: ['./esop-page.component.scss']

})

export class EsopPageComponent implements OnInit {

  user: User | null = null;

  esopSymbols: Symbol[] = [];

  totalStocksGranted: number = 0;

  totalVestedStocks: number = 0;

  totalUnvestedStocks: number = 0;

  currentMarketValue: number = 0;

  nextVestingDate: string = '';

  timeToFullVesting: { months: number, days: number } = { months: 0, days: 0 };

  currentStockPrice: number = 100; // Default value

  discountedPrice: number = 80; // Default value

  vestingPercentage: number = 0;

  exercisedStocks: number = 0;

  remainingStocksToExercise: number = 0;

  constructor(

    private authService: AuthService,

    private calculationService: CalculationService

  ) {}

  ngOnInit(): void {

    this.user = this.authService.currentUserValue;

    if (!this.user) return;

    // Find ESOP awards

    const esopAward = this.user.awards.find(award => award.type === 'esop');

    if (esopAward) {

      this.esopSymbols = esopAward.symbols;

      // Calculate totals

      this.calculateTotals();

    }

  }

  calculateTotals(): void {

    this.totalStocksGranted = this.esopSymbols.reduce((total, symbol) => total + symbol.granted, 0);

    this.totalVestedStocks = this.esopSymbols.reduce((total, symbol) => total + (symbol.vested || 0), 0);

    this.totalUnvestedStocks = this.totalStocksGranted - this.totalVestedStocks;

    // Get discounted price from first symbol if available

    if (this.esopSymbols.length > 0 && this.esopSymbols[0].discounted\_price) {

      this.discountedPrice = this.esopSymbols[0].discounted\_price;

    }

    // Calculate current market value

    this.currentMarketValue = this.esopSymbols.reduce((total, symbol) => {

      return total + this.calculationService.calculateMarketValue(symbol);

    }, 0);

    // Find next vesting date

    const nextDates = this.esopSymbols

      .map(symbol => symbol.next\_vesting\_date)

      .filter(date => !!date)

      .sort();

    if (nextDates.length > 0) {

      this.nextVestingDate = this.calculationService.formatNextVestingDate(nextDates[0]);

    }

    // Calculate time to full vesting

    if (this.esopSymbols.length > 0 && this.esopSymbols[0].next\_vesting\_date) {

      this.timeToFullVesting = this.calculationService.calculateTimeToFullVesting(this.esopSymbols[0]);

    }

    // Calculate vesting percentage

    if (this.totalStocksGranted > 0) {

      this.vestingPercentage = Math.round((this.totalVestedStocks / this.totalStocksGranted) \* 100);

    }

    // Calculate exercised stocks and remaining stocks to exercise

    this.exercisedStocks = this.esopSymbols.reduce((total, symbol) => total + (symbol.exercisable || 0), 0);

    this.remainingStocksToExercise = this.totalVestedStocks - this.exercisedStocks;

  }

}

Rsu-page.html

<!-- src/app/components/rsu-page/rsu-page.component.html -->

<div class="rsu-container">

    <div class="header">

      <div class="title">

        <h1>RSU</h1>

      </div>

      <div class="stock-price">Current Stock Price: <span class="price-value">${{ currentStockPrice }}</span></div>

      <div class="user-info">

        <span>{{ user?.name }}</span>

        <img src="assets/images/user-avatar.png" alt="User Avatar" class="avatar">

      </div>

    </div>

    <div class="data-grid">

      <div class="data-card">

        <h3>Total Stocks Granted</h3>

        <div class="card-value">{{ totalStocksGranted }}</div>

      </div>

      <div class="data-card">

        <h3>Total Vested Stocks</h3>

        <div class="card-value">{{ totalVestedStocks }}</div>

      </div>

      <div class="data-card">

        <h3>Remaining Unvested Stocks</h3>

        <div class="card-value">{{ totalUnvestedStocks }}</div>

      </div>

      <div class="data-card">

        <h3>My Current Value</h3>

        <div class="card-value">${{ currentValue }}</div>

      </div>

      <div class="data-card">

        <h3>Next Vesting Date</h3>

        <div class="card-value">{{ nextVestingDate }}</div>

      </div>

    </div>

    <div class="widget-grid">

      <div class="widget">

        <h3>Grant Agreements</h3>

        <button class="download-btn">Download</button>

      </div>

      <div class="widget">

        <h3>Vesting Progress</h3>

        <div class="progress-circle">

          <svg viewBox="0 0 36 36" class="circular-chart">

            <path class="circle-bg"

              d="M18 2.0845

                a 15.9155 15.9155 0 0 1 0 31.831

                a 15.9155 15.9155 0 0 1 0 -31.831"

            />

            <path class="circle"

              [attr.stroke-dasharray]="vestingPercentage + ', 100'"

              d="M18 2.0845

                a 15.9155 15.9155 0 0 1 0 31.831

                a 15.9155 15.9155 0 0 1 0 -31.831"

            />

            <text x="18" y="20.35" class="percentage">{{ vestingPercentage }}%</text>

          </svg>

          <div class="vested-label">Vested</div>

        </div>

      </div>

      <div class="widget">

        <h3>Vesting History</h3>

        <div class="vesting-history">

          <div class="history-item">

            <span class="history-label">1st Vest:</span>

            <span class="history-value">{{ firstVestingDate }}</span>

          </div>

        </div>

      </div>

      <div class="widget">

        <h3>Time to Full Vesting</h3>

        <div class="time-to-vesting">

          <span class="time-value">{{ timeToFullVesting.months }}</span>

          <span class="time-label">months,</span>

          <span class="time-value">{{ timeToFullVesting.days }}</span>

          <span class="time-label">days</span>

        </div>

      </div>

      <div class="widget">

        <h3>Tax Calculator</h3>

        <div class="tax-calculator">

          <div class="calculator-fields">

            <div class="field">

              <span>State</span>

            </div>

            <div class="field">

              <span>Rate</span>

            </div>

            <div class="field">

              <span>Total</span>

            </div>

          </div>

          <button class="calculate-btn">Calculate</button>

        </div>

      </div>

    </div>

  </div>

Rsu-page.ts

// src/app/components/rsu-page/rsu-page.component.ts

import { Component, OnInit } from '@angular/core';

import { CommonModule } from '@angular/common';

import { AuthService } from '../../services/auth.service';

import { CalculationService } from '../../services/calculation.service';

import { User, Symbol } from '../../models/user';

@Component({

  selector: 'app-rsu-page',

  standalone: true,

  imports: [CommonModule],

  templateUrl: './rsu-page.component.html',

  styleUrls: ['./rsu-page.component.scss']

})

export class RsuPageComponent implements OnInit {

  user: User | null = null;

  rsuSymbols: Symbol[] = [];

  totalStocksGranted: number = 0;

  totalVestedStocks: number = 0;

  totalUnvestedStocks: number = 0;

  currentValue: number = 0;

  nextVestingDate: string = '';

  timeToFullVesting: { months: number, days: number } = { months: 0, days: 0 };

  currentStockPrice: number = 100; // Default value

  vestingPercentage: number = 0;

  firstVestingDate: string = '';

  constructor(

    private authService: AuthService,

    private calculationService: CalculationService

  ) {}

  ngOnInit(): void {

    this.user = this.authService.currentUserValue;

    if (!this.user) return;

    // Find RSU awards

    const rsuAward = this.user.awards.find(award => award.type === 'rsu');

    if (rsuAward) {

      this.rsuSymbols = rsuAward.symbols;

      // Calculate totals

      this.calculateTotals();

    }

  }

  calculateTotals(): void {

    this.totalStocksGranted = this.rsuSymbols.reduce((total, symbol) => total + symbol.granted, 0);

    this.totalVestedStocks = this.rsuSymbols.reduce((total, symbol) => total + (symbol.vested || 0), 0);

    this.totalUnvestedStocks = this.totalStocksGranted - this.totalVestedStocks;

    // Calculate current value

    this.currentValue = this.rsuSymbols.reduce((total, symbol) => {

      return total + this.calculationService.calculateMarketValue(symbol);

    }, 0);

    // Find next vesting date

    const nextDates = this.rsuSymbols

      .map(symbol => symbol.next\_vesting\_date)

      .filter(date => !!date)

      .sort();

    if (nextDates.length > 0) {

      this.nextVestingDate = this.calculationService.formatNextVestingDate(nextDates[0]);

      // Find first symbol with this vesting date

      const symbolWithNextVesting = this.rsuSymbols.find(

        symbol => symbol.next\_vesting\_date === nextDates[0]

      );

      if (symbolWithNextVesting) {

        this.timeToFullVesting = this.calculationService.calculateTimeToFullVesting(symbolWithNextVesting);

      }

    }

    // Calculate vesting percentage

    if (this.totalStocksGranted > 0) {

      this.vestingPercentage = Math.round((this.totalVestedStocks / this.totalStocksGranted) \* 100);

    }

    // Find first vesting date

    if (this.rsuSymbols.length > 0 && this.rsuSymbols[0].vesting\_schedule && this.rsuSymbols[0].vesting\_schedule.length > 0) {

      this.firstVestingDate = this.calculationService.formatNextVestingDate(this.rsuSymbols[0].vesting\_schedule[0].date);

    }

  }

}

Esop-page.html

<!-- src/app/components/esop-page/esop-page.component.html -->

<div class="esop-container">

    <div class="header">

      <div class="title">

        <h1>ESOP</h1>

      </div>

      <div class="stock-prices">

        <div class="price-item">Current Stock Price: <span class="price-value">${{ currentStockPrice }}</span></div>

        <div class="price-item">Discounted Price: <span class="price-value discounted">${{ discountedPrice }}</span></div>

      </div>

      <div class="user-info">

        <span>{{ user?.name }}</span>

        <img src="assets/images/user-avatar.png" alt="User Avatar" class="avatar">

      </div>

    </div>

    <div class="data-grid">

      <div class="data-card">

        <h3>Total Stocks Granted</h3>

        <div class="card-value">{{ totalStocksGranted }}</div>

      </div>

      <div class="data-card">

        <h3>Total Vested Stocks</h3>

        <div class="card-value">{{ totalVestedStocks }}</div>

      </div>

      <div class="data-card">

        <h3>Remaining Unvested Stocks</h3>

        <div class="card-value">{{ totalUnvestedStocks }}</div>

      </div>

      <div class="data-card">

        <h3>Current Market Value</h3>

        <div class="card-value">${{ currentMarketValue }}</div>

      </div>

      <div class="data-card">

        <h3>Next Vesting Date</h3>

        <div class="card-value">{{ nextVestingDate }}</div>

      </div>

    </div>

    <div class="widget-grid">

      <div class="data-card">

        <h3>Time to Full Vesting</h3>

        <div class="time-to-vesting">

          <span class="time-value">{{ timeToFullVesting.months }}</span>

          <span class="time-label">months,</span>

          <span class="time-value">{{ timeToFullVesting.days }}</span>

          <span class="time-label">days</span>

        </div>

      </div>

      <div class="data-card">

        <h3>Vesting Progress</h3>

        <div class="progress-circle">

          <svg viewBox="0 0 36 36" class="circular-chart">

            <path class="circle-bg"

              d="M18 2.0845

                a 15.9155 15.9155 0 0 1 0 31.831

                a 15.9155 15.9155 0 0 1 0 -31.831"

            />

            <path class="circle"

              [attr.stroke-dasharray]="vestingPercentage + ', 100'"

              d="M18 2.0845

                a 15.9155 15.9155 0 0 1 0 31.831

                a 15.9155 15.9155 0 0 1 0 -31.831"

            />

            <text x="18" y="20.35" class="percentage">{{ vestingPercentage }}%</text>

          </svg>

          <div class="vested-label">Vested</div>

        </div>

      </div>

      <div class="data-card">

        <h3>Exercised Stocks</h3>

        <div class="card-value">{{ exercisedStocks }}</div>

      </div>

      <div class="data-card">

        <h3>Remaining Stocks to Exercise</h3>

        <div class="card-value">{{ remainingStocksToExercise }}</div>

      </div>

      <div class="data-card">

        <h3>Tax Calculator</h3>

        <div class="tax-calculator">

          <div class="calculator-fields">

            <div class="field">

              <span>State</span>

            </div>

            <div class="field">

              <span>Rate</span>

            </div>

            <div class="field">

              <span>Total</span>

            </div>

          </div>

          <button class="calculate-btn">Calculate</button>

        </div>

      </div>

    </div>

  </div>

Esop-page.ts

// src/app/components/esop-page/esop-page.component.ts

import { Component, OnInit } from '@angular/core';

import { CommonModule } from '@angular/common';

import { AuthService } from '../../services/auth.service';

import { CalculationService } from '../../services/calculation.service';

import { User, Symbol } from '../../models/user';

@Component({

  selector: 'app-esop-page',

  standalone: true,

  imports: [CommonModule],

  templateUrl: './esop-page.component.html',

  styleUrls: ['./esop-page.component.scss']

})

export class EsopPageComponent implements OnInit {

  user: User | null = null;

  esopSymbols: Symbol[] = [];

  totalStocksGranted: number = 0;

  totalVestedStocks: number = 0;

  totalUnvestedStocks: number = 0;

  currentMarketValue: number = 0;

  nextVestingDate: string = '';

  timeToFullVesting: { months: number, days: number } = { months: 0, days: 0 };

  currentStockPrice: number = 100; // Default value

  discountedPrice: number = 80; // Default value

  vestingPercentage: number = 0;

  exercisedStocks: number = 0;

  remainingStocksToExercise: number = 0;

  constructor(

    private authService: AuthService,

    private calculationService: CalculationService

  ) {}

  ngOnInit(): void {

    this.user = this.authService.currentUserValue;

    if (!this.user) return;

    // Find ESOP awards

    const esopAward = this.user.awards.find(award => award.type === 'esop');

    if (esopAward) {

      this.esopSymbols = esopAward.symbols;

      // Calculate totals

      this.calculateTotals();

    }

  }

  calculateTotals(): void {

    this.totalStocksGranted = this.esopSymbols.reduce((total, symbol) => total + symbol.granted, 0);

    this.totalVestedStocks = this.esopSymbols.reduce((total, symbol) => total + (symbol.vested || 0), 0);

    this.totalUnvestedStocks = this.totalStocksGranted - this.totalVestedStocks;

    // Get discounted price from first symbol if available

    if (this.esopSymbols.length > 0 && this.esopSymbols[0].discounted\_price) {

      this.discountedPrice = this.esopSymbols[0].discounted\_price;

    }

    // Calculate current market value

    this.currentMarketValue = this.esopSymbols.reduce((total, symbol) => {

      return total + this.calculationService.calculateMarketValue(symbol);

    }, 0);

    // Find next vesting date

    const nextDates = this.esopSymbols

      .map(symbol => symbol.next\_vesting\_date)

      .filter(date => !!date)

      .sort();

    if (nextDates.length > 0) {

      this.nextVestingDate = this.calculationService.formatNextVestingDate(nextDates[0]);

    }

    // Calculate time to full vesting

    if (this.esopSymbols.length > 0 && this.esopSymbols[0].next\_vesting\_date) {

      this.timeToFullVesting = this.calculationService.calculateTimeToFullVesting(this.esopSymbols[0]);

    }

    // Calculate vesting percentage

    if (this.totalStocksGranted > 0) {

      this.vestingPercentage = Math.round((this.totalVestedStocks / this.totalStocksGranted) \* 100);

    }

    // Calculate exercised stocks and remaining stocks to exercise

    this.exercisedStocks = this.esopSymbols.reduce((total, symbol) => total + (symbol.exercisable || 0), 0);

    this.remainingStocksToExercise = this.totalVestedStocks - this.exercisedStocks;

  }

}

Vest-page.html

<div class="vest-container">

    <div class="header">

      <h1>VESTING SCHEDULE</h1>

      <div class="subtitle">The gains at which equity was granted</div>

      <div class="user-name">{{userData?.name}}</div>

    </div>

    <div class="table-container">

      <table class="vesting-table">

        <thead>

          <tr>

            <th>Sl No</th>

            <th>Symbol</th>

            <th>Type</th>

            <th>Award Price</th>

            <th>Granted</th>

            <th>Exercisable</th>

            <th>Final Exercise Date</th>

            <th>Exercisable Market Value</th>

            <th>Status</th>

          </tr>

        </thead>

        <tbody>

          <tr \*ngFor="let item of vestingSchedule">

            <td>{{item.id}}</td>

            <td>{{item.symbol}}</td>

            <td>{{item.type}}</td>

            <td>{{item.award\_price}}</td>

            <td>{{item.granted}}</td>

            <td>{{item.exercisable}}</td>

            <td>{{item.final\_exercise\_date}}</td>

            <td>{{item.exercisable\_market\_value}}</td>

            <td>

              <span class="status-badge" [ngClass]="{'complete': item.status === 'complete', 'pending': item.status === 'pending'}">

                {{item.status}}

              </span>

            </td>

          </tr>

        </tbody>

      </table>

    </div>

    <div class="pagination">

      <button class="pagination-btn prev">

        <span class="arrow">‹</span>

      </button>

      <button class="pagination-btn next">

        <span class="arrow">›</span>

      </button>

    </div>

  </div>

  <div \*ngIf="!vestingSchedule.length" class="no-data">

    <p>No vesting schedule available for this user.</p>

  </div>

Vest-page.ts

import { Component, OnInit } from '@angular/core';

import { CommonModule } from '@angular/common';

import { UserService } from '../../services/user.service';

import { CalculationService } from '../../services/calculation.service';

import { User } from '../../models/user';

import { VestingScheduleItem } from '../../models/award';

@Component({

  selector: 'app-vest-page',

  standalone: true,

  imports: [CommonModule],

  templateUrl: './vest-page.component.html',

  styleUrls: ['./vest-page.component.scss']

})

export class VestPageComponent implements OnInit {

  userData: User | null = null;

  vestingSchedule: VestingScheduleItem[] = [];

  constructor(

    private userService: UserService,

    private calculationService: CalculationService

  ) {}

  ngOnInit(): void {

    this.userService.getUserData().subscribe(userData => {

      this.userData = userData;

      if (userData) {

        this.vestingSchedule = this.calculationService.calculateVestingSchedule(userData);

      }

    });

  }

}

User.ts model

// src/app/models/user.model.ts

export interface UserTaxes {

    federal\_tax\_rate: number;

    state\_tax\_rate: number;

    local\_tax\_rate: number;

  }

  export interface Symbol {

    symbol: string;

    award\_price: number;

    granted: number;

    grant\_date: string;

    exercisable?: number;

    discounted\_price?: number;

    vested?: number;

    unvested?: number;

    next\_vesting\_date?: string;

    vesting\_schedule?: VestingScheduleItem[];

  }

  export interface VestingScheduleItem {

    date: string;

    amount: number;

    status: 'pending' | 'complete';

  }

  export interface Award {

    type: 'rsu' | 'esop';

    symbols: Symbol[];

  }

  export interface User {

    emp\_id: string;

    name: string;

    email: string;

    pwd: string;

    taxes: UserTaxes;

    awards: Award[];

  }

Calculation.service.ts

// src/app/services/calculation.service.ts

import { Injectable } from '@angular/core';

import { Symbol, User } from '../models/user';

@Injectable({

  providedIn: 'root'

})

export class CalculationService {

  // Mock current stock prices - in a real app, these would come from an API

  private currentStockPrices: Record<string, number> = {

    'AAPL': 100,

    'GOOG': 100,

    'MSFT': 100,

    'AMZN': 100,

    'IBM': 100

  };

  constructor() { }

  getCurrentPrice(symbol: string): number {

    return this.currentStockPrices[symbol] || 100; // Default to 100 if not found

  }

  getVestedAmount(symbol: Symbol): number {

    return symbol.vested || 0;

  }

  getUnvestedAmount(symbol: Symbol): number {

    return symbol.unvested || 0;

  }

  // Calculate current market value based on vested shares and current price

  calculateMarketValue(symbol: Symbol): number {

    const currentPrice = this.getCurrentPrice(symbol.symbol);

    const vestedShares = this.getVestedAmount(symbol);

    return currentPrice \* vestedShares;

  }

  // Calculate total exercisable value

  calculateExercisableValue(symbol: Symbol): number {

    const currentPrice = this.getCurrentPrice(symbol.symbol);

    const exercisable = symbol.exercisable || 0;

    return currentPrice \* exercisable;

  }

  // Calculate time to full vesting in months and days

  calculateTimeToFullVesting(symbol: Symbol): { months: number, days: number } {

    if (!symbol.next\_vesting\_date) {

      return { months: 0, days: 0 };

    }

    const nextVestingDate = new Date(symbol.next\_vesting\_date);

    const currentDate = new Date();

    // Get total days between dates

    const diffTime = Math.abs(nextVestingDate.getTime() - currentDate.getTime());

    const diffDays = Math.ceil(diffTime / (1000 \* 60 \* 60 \* 24));

    const months = Math.floor(diffDays / 30);

    const days = diffDays % 30;

    return { months, days };

  }

  // Format the next vesting date (e.g., Aug 30, 2025)

  formatNextVestingDate(dateString?: string): string {

    if (!dateString) return 'N/A';

    const date = new Date(dateString);

    const month = date.toLocaleString('default', { month: 'short' });

    const day = date.getDate();

    const year = date.getFullYear();

    return `${month} ${day}, ${year}`;

  }

  // Calculate vesting percentage

  calculateVestingPercentage(symbol: Symbol): number {

    if (!symbol.granted || symbol.granted === 0) return 0;

    return Math.round((this.getVestedAmount(symbol) / symbol.granted) \* 100);

  }

  // Calculate tax for a given award

  calculateTax(value: number, user: User): number {

    const totalTaxRate = (user.taxes.federal\_tax\_rate +

                          user.taxes.state\_tax\_rate +

                          user.taxes.local\_tax\_rate) / 100;

    return value \* totalTaxRate;

  }

}