# **Product Requirements Document**

## **CheckDisOut - Achievement Aggregator Web App for Coders**

## **1. Project Overview & Goals**

### **Project Description**

**CheckDisOut is an Achievement Aggregator Web App for coders - primarily students - who participate in various competitions/hackathons and wish to showcase their participations, achievements, or just personal projects that they have made, which also helps the users export their portfolio to be added at a prominent place as proof of work or just to show off their skills.**

### **Problem Statement**

Students participate in many competitions but can't showcase them easily. Build: A dashboard where students can store and showcase their projects, event participation, certificates, and achievements -- exportable as a public profile or portfolio.

### **Target Audience**

* Students
* Hackathon Enthusiasts
* Junior Developers

### **Success Metrics/Goals**

Seamless experience of adding achievements/certifications/projects to one's profile and exporting it in the form of a PDF with active links within the PDF, which can be used for validation purposes.

## **2. Technical Architecture**

Frontend: NextJS (App Router) + TailwindCSS

Backend: Firebase (Firestore + Auth + Storage)

Icons: Lucide React

Database: Firestore

Authentication: Firebase Auth (Email/Password + Google OAuth)

Deployment: Vercel

PDF Generation: React-PDF (@react-pdf/renderer)

File Upload: Firebase Storage

State Management: React Context + useState

Fonts: Space Grotesk (headings), Inter (content)

### **Environment Variables Required**

NEXT\_PUBLIC\_FIREBASE\_API\_KEY=

NEXT\_PUBLIC\_FIREBASE\_AUTH\_DOMAIN=

NEXT\_PUBLIC\_FIREBASE\_PROJECT\_ID=

NEXT\_PUBLIC\_FIREBASE\_STORAGE\_BUCKET=

NEXT\_PUBLIC\_FIREBASE\_MESSAGING\_SENDER\_ID=

NEXT\_PUBLIC\_FIREBASE\_APP\_ID=

NEXT\_PUBLIC\_BASE\_URL=

## **3. Database Schema Design**

### **Firestore Collections Structure**

// users collection

users: {

uid: string,

email: string,

name: string,

age: number,

gender: 'male' | 'female',

profilePicture: string, // Firebase Storage URL

isStudent: boolean, // true for student, false for professional

qualities: string[], // Array of 5 qualities

workExperience: string,

education: string,

skills: string[], // Array of skills

socials: {

github?: string,

linkedin?: string,

discord?: string,

instagram?: string,

portfolio?: string

},

createdAt: timestamp,

updatedAt: timestamp

}

// achievements collection

achievements: {

id: string,

userId: string,

competitionName: string,

organizingBody: string,

position: string,

competitionDate: date,

isOnline: boolean,

eventLink?: string, // if online

location?: string, // if offline

isSolo: boolean,

teamMembers?: [{

name: string,

socials: {

github?: string,

linkedin?: string,

discord?: string

}

}],

certificateUrl?: string, // Firebase Storage URL

githubLink?: string,

createdAt: timestamp,

updatedAt: timestamp

}

// participations collection

participations: {

id: string,

userId: string,

competitionName: string,

organizingBody: string,

competitionDate: date,

isOnline: boolean,

eventLink?: string,

location?: string,

isSolo: boolean,

teamMembers?: [{

name: string,

socials: {

github?: string,

linkedin?: string,

discord?: string

}

}],

certificateUrl?: string,

githubLink?: string,

createdAt: timestamp,

updatedAt: timestamp

}

// projects collection

projects: {

id: string,

userId: string,

projectName: string,

description: string,

bannerImageUrl: string, // Firebase Storage URL

githubLink: string,

isSolo: boolean,

teamMembers?: [{

name: string,

socials: {

github?: string,

linkedin?: string,

discord?: string

}

}],

createdAt: timestamp,

updatedAt: timestamp

}

## **4. File & Folder Structure**

app/

├── (auth)/

│ ├── login/

│ │ └── page.tsx

│ ├── register/

│ │ └── page.tsx

│ └── layout.tsx

├── (dashboard)/

│ ├── home/

│ │ └── page.tsx

│ ├── achievements/

│ │ └── page.tsx

│ ├── participations/

│ │ └── page.tsx

│ ├── projects/

│ │ └── page.tsx

│ ├── export-portfolio/

│ │ └── page.tsx

│ ├── add-to-portfolio/

│ │ └── page.tsx

│ ├── profile/

│ │ └── page.tsx

│ └── layout.tsx

├── api/

│ ├── generate-pdf/

│ │ └── route.ts

│ └── upload/

│ └── route.ts

├── globals.css

└── layout.tsx

components/

├── ui/

│ ├── Button.tsx

│ ├── Card.tsx

│ ├── Input.tsx

│ ├── Modal.tsx

│ ├── Dropdown.tsx

│ └── FileUpload.tsx

├── layout/

│ ├── Navbar.tsx

│ └── AuthGuard.tsx

├── forms/

│ ├── AchievementForm.tsx

│ ├── ParticipationForm.tsx

│ ├── ProjectForm.tsx

│ └── ProfileForm.tsx

├── cards/

│ ├── AchievementCard.tsx

│ ├── ParticipationCard.tsx

│ └── ProjectCard.tsx

├── pdf/

│ └── PortfolioPDF.tsx

└── animations/

└── TextAnimation.tsx

lib/

├── firebase/

│ ├── config.ts

│ ├── auth.ts

│ ├── firestore.ts

│ └── storage.ts

├── context/

│ ├── AuthContext.tsx

│ └── UserDataContext.tsx

├── hooks/

│ ├── useAuth.ts

│ ├── useFirestore.ts

│ └── useStorage.ts

├── utils/

│ ├── validation.ts

│ ├── constants.ts

│ └── helpers.ts

└── types/

└── index.ts

public/

├── logo.png

└── placeholder-avatar.png

## **5. Styling Guidelines**

### **Fonts**

/\* Import in globals.css \*/

@import url('https://fonts.googleapis.com/css2?family=Space+Grotesk:wght@300;400;500;600;700&display=swap');

@import url('https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap');

/\* Usage \*/

.font-heading { font-family: 'Space Grotesk', sans-serif; }

.font-body { font-family: 'Inter', sans-serif; }

### **Color Palette (Exact Hex Codes)**

const colors = {

eerieBlack: '#1D1E18', // navbar background, card background

honeydew: '#F1FFE7', // font over dark background

vividSkyBlue: '#00D4FF', // dividers, buttons, shadows, text shadow

darkPurple: '#300C3B', // button backgrounds, horizontal bars

mauveine: '#791E94', // dropdown backgrounds, button shadows

grayReset: '#474747', // reset button background

dangerRed: '#FF0004', // delete button

successGreen: '#00FF1E', // accept buttons

// Additional utility colors

cardBorder: '#2D2E28',

inputBg: '#2A2B25',

textMuted: '#A8A8A8'

}

### **TailwindCSS Custom Configuration**

// tailwind.config.js additions

module.exports = {

theme: {

extend: {

colors: {

'eerie-black': '#1D1E18',

'honeydew': '#F1FFE7',

'vivid-sky-blue': '#00D4FF',

'dark-purple': '#300C3B',

'mauveine': '#791E94',

'gray-reset': '#474747',

'danger-red': '#FF0004',

'success-green': '#00FF1E'

},

fontFamily: {

'heading': ['Space Grotesk', 'sans-serif'],

'body': ['Inter', 'sans-serif']

},

animation: {

'slide-up': 'slideUp 1s ease-out',

'fade-in': 'fadeIn 0.8s ease-in',

'text-cycle': 'textCycle 8s infinite'

}

}

}

}

## **6. Component Specifications**

### **6.1 Navbar Component**

// Location: components/layout/Navbar.tsx

// Vertical sidebar navigation

// Fixed position, full height

// Dark background (#1D1E18)

// Logo at top: "CDO" in Space Grotesk font

// Navigation items with Lucide icons:

// - Home (Home icon)

// - Achievements (Trophy icon)

// - Participations (Users icon)

// - Projects (FolderOpen icon)

// - Export Portfolio (Download icon)

// - Add to Portfolio (Plus icon)

// - Profile (User icon)

// - Logout (LogOut icon)

### **6.2 Authentication Components**

// Login/Register forms

// Email/password + Google OAuth

// Form validation with error states

// Redirect to /home after successful auth

// Styled with dark theme

### **6.3 Home Page Components**

// components/pages/HomePage.tsx

// Layout: Profile section + Recent achievements + Action buttons

// Animations:

// - Name slides up from purple bar on page load

// - Age slides up with delay

// - Qualities rotate every 2 seconds (4 qualities cycle)

// Recent achievements: Top 3 by date

// Action buttons link to Projects and Participations

### **6.4 Card Components**

// AchievementCard.tsx & ParticipationCard.tsx

// Grid: 2 cards per row on desktop, 1 on mobile

// Each card contains all specified fields

// 3 action buttons: Delete (red), Edit (blue), Share (green)

// Edit functionality: inline editing with save/cancel

// Share functionality: copy link to clipboard with toast

### **6.5 Form Components**

// Dynamic forms for Add to Portfolio page

// Tab navigation between Achievement/Participation/Project

// Conditional fields based on selections (online/offline, solo/team)

// File upload for certificates and project banners

// Form validation with real-time feedback

// Reset functionality clears all fields

// Success page with options to add more or go home

## **7. Page Specifications**

### **7.1 Authentication Flow**

Route: /login, /register

Layout: Centered form with logo

Redirect: Authenticated users to /home

Features: Email/password, Google OAuth

Validation: Real-time form validation

### **7.2 Home Page (/home)**

// Layout sections:

// 1. Hero section with profile info and animated elements

// 2. Skills section with pill-style tags

// 3. Recent achievements (top 3)

// 4. Action buttons section

// Responsive: Single column on mobile, two-column on desktop

### **7.3 Achievements Page (/achievements)**

// Grid layout: 2 cards per row

// Each card shows all achievement data

// Action buttons: Delete, Edit, Share

// Loading states for data fetching

// Empty state when no achievements

// Pagination if more than 10 items

### **7.4 Participations Page (/participations)**

// Similar to achievements but without "position" field

// Same card layout and functionality

### **7.5 Projects Page (/projects)**

// Single column layout (1 card per row)

// Larger cards with project banner images

// Description with read more/less functionality

// GitHub link opens in new tab

### **7.6 Export Portfolio Page (/export-portfolio)**

// Single card layout

// Checkbox list for sections to include

// State management:

// - Default: all sections selected (green)

// - User can toggle with X/✓ buttons

// - Confirm button enables Export button

// - Export generates PDF and triggers download

### **7.7 Add to Portfolio Page (/add-to-portfolio)**

// Tab navigation: Achievement | Participation | Project

// Forms with conditional fields

// File upload areas with drag & drop

// Team member dynamic fields (add/remove)

// Form validation and error handling

// Success flow: confirmation → add more or go home

### **7.8 Profile Page (/profile)**

// User info at top with profile picture

// Dropdown sections in alternating colors (purple/blue)

// Edit mode: inline editing for allowed fields

// Export button links to export page

// Data sections are read-only (populated from other forms)

## **8. State Management Strategy**

### **8.1 Authentication Context**

// AuthContext provides:

// - user object

// - login/logout functions

// - loading states

// - authentication status

### **8.2 User Data Context**

// UserDataContext provides:

// - user profile data

// - achievements array

// - participations array

// - projects array

// - loading states

// - CRUD operations for each data type

### **8.3 Component State**

// Local state for:

// - Form data and validation

// - Modal open/close states

// - Loading states for individual actions

// - Edit mode toggles

## **9. Firebase Integration Specifics**

### **9.1 Authentication Setup**

// Firebase Auth methods:

// - createUserWithEmailAndPassword

// - signInWithEmailAndPassword

// - signInWithPopup (Google)

// - signOut

// - onAuthStateChanged listener

### **9.2 Firestore Security Rules**

rules\_version = '2';

service cloud.firestore {

match /databases/{database}/documents {

// Users can only access their own data

match /users/{userId} {

allow read, write: if request.auth != null && request.auth.uid == userId;

}

match /achievements/{achievementId} {

allow read, write: if request.auth != null && request.auth.uid == resource.data.userId;

}

match /participations/{participationId} {

allow read, write: if request.auth != null && request.auth.uid == resource.data.userId;

}

match /projects/{projectId} {

allow read, write: if request.auth != null && request.auth.uid == resource.data.userId;

}

}

}

### **9.3 Storage Setup**

// Firebase Storage rules

rules\_version = '2';

service firebase.storage {

match /b/{bucket}/o {

match /users/{userId}/{allPaths=\*\*} {

allow read, write: if request.auth != null && request.auth.uid == userId;

}

}

}

## **10. Icons & Visual Assets**

### **Icon Library: Lucide React**

// Key icons needed:

import {

Home, Trophy, Users, FolderOpen, Download, Plus, User, LogOut,

Edit, Trash2, Share, Github, Linkedin, Mail, MapPin, Calendar,

ExternalLink, Upload, Check, X, ChevronDown, ChevronRight,

Camera, Save, RefreshCw, FileText, Globe, Instagram, MessageCircle

} from 'lucide-react';

// Sizing conventions:

// Small: w-4 h-4 (16px)

// Medium: w-5 h-5 (20px)

// Large: w-6 h-6 (24px)

// XL: w-8 h-8 (32px)

// Color patterns:

// Default: text-gray-400

// Primary: text-vivid-sky-blue

// Danger: text-danger-red

// Success: text-success-green

// Hover states: hover:text-[color]-600

## **11. Animation Specifications**

### **11.1 Home Page Animations**

/\* Name slide-up animation \*/

@keyframes slideUp {

from { transform: translateY(50px); opacity: 0; }

to { transform: translateY(0); opacity: 1; }

}

/\* Qualities rotation animation \*/

@keyframes textCycle {

0%, 20% { opacity: 1; }

25%, 95% { opacity: 0; }

100% { opacity: 1; }

}

### **11.2 Component Animations**

// Card hover animations

// Button click animations

// Loading spinners

// Fade in/out for modals

// Slide transitions for forms

## **12. PDF Export Specifications**

### **12.1 PDF Generation Component**

// components/pdf/PortfolioPDF.tsx

// Uses React-PDF library

// Includes selected sections based on user choice

// Styling matches web app theme

// Clickable links preserved in PDF

// Professional layout with proper spacing

### **12.2 Export API Route**

// app/api/generate-pdf/route.ts

// Handles PDF generation server-side if needed

// Returns PDF blob for download

// Error handling for generation failures

## **13. Error Handling & Validation**

### **13.1 Form Validation**

// Real-time validation for all forms

// Required field indicators

// Format validation (emails, URLs)

// File type/size validation

// Error messages with clear instructions

### **13.2 Error Boundaries**

// React Error Boundaries for graceful failures

// Fallback UI for component errors

// Error logging for debugging

### **13.3 Loading States**

// Skeleton loaders for data fetching

// Button loading states

// Page-level loading indicators

// Empty states with helpful messages

## **14. Responsive Design Breakpoints**

/\* Mobile First Approach \*/

/\* Default: Mobile (< 640px) \*/

/\* sm: Tablet (640px+) \*/

/\* md: Small Desktop (768px+) \*/

/\* lg: Desktop (1024px+) \*/

/\* xl: Large Desktop (1280px+) \*/

/\* Specific responsive behaviors: \*/

/\* - Navbar: Collapsed on mobile, expanded on desktop \*/

/\* - Cards: 1 column mobile, 2 columns desktop \*/

/\* - Forms: Full width mobile, centered desktop \*/

/\* - Profile dropdowns: Stack on mobile \*/

## **15. Performance Optimization**

### **15.1 Image Optimization**

// Next.js Image component for all images

// Lazy loading for cards

// WebP format conversion

// Responsive image sizing

### **15.2 Code Splitting**

// Dynamic imports for heavy components

// Route-based code splitting

// Lazy loading for PDF generation

### **15.3 Firebase Optimization**

// Pagination for large datasets

// Real-time listeners cleanup

// Optimistic updates for better UX

// Image compression before upload

## **16. Testing Strategy**

### **16.1 User Scenarios to Test**

1. User registration and login flow

2. Adding achievement/participation/project

3. Editing existing items

4. Deleting items with confirmation

5. Profile editing and saving

6. PDF export with different selections

7. File uploads (certificates, images)

8. Social links validation

9. Responsive behavior on mobile

10. Error handling for network failures

### **16.2 Edge Cases**

- Empty states (no achievements, etc.)

- Large file uploads

- Network connectivity issues

- Invalid URLs in social links

- Duplicate form submissions

- Expired authentication tokens

## **17. Deployment Configuration**

### **17.1 Vercel Deployment**

// vercel.json

{

"framework": "nextjs",

"buildCommand": "npm run build",

"devCommand": "npm run dev",

"installCommand": "npm install",

"functions": {

"app/api/generate-pdf/route.ts": {

"maxDuration": 30

}

}

}

### **17.2 Environment Setup**

# Firebase project setup

npm install firebase

firebase login

firebase init

# Vercel deployment

npm install -g vercel

vercel --prod

## **18. Priority Implementation Order**

### **Phase 1 (MVP - Core Functionality)**

1. Authentication system (login/register)
2. User profile creation
3. Basic navbar navigation
4. Add Achievement form
5. View Achievements page
6. Basic PDF export

### **Phase 2 (Extended Features)**

1. Add Participation and Project forms
2. Edit/Delete functionality
3. Profile page with dropdowns
4. Home page with animations
5. Enhanced PDF export options

### **Phase 3 (Polish & Optimization)**

1. Animations and micro-interactions
2. Advanced validation
3. Error handling improvements
4. Performance optimizations
5. Mobile responsiveness refinements

## **19. Success Criteria**

### **Technical Success**

* [ ] All forms working with validation
* [ ] PDF export functioning with active links
* [ ] Authentication flow complete
* [ ] Data persistence in Firestore
* [ ] Responsive on all devices
* [ ] No console errors
* [ ] Loading states implemented

### **User Experience Success**

* [ ] Intuitive navigation
* [ ] Smooth animations
* [ ] Fast page loads (< 3 seconds)
* [ ] Professional PDF output
* [ ] Clear error messages
* [ ] Accessible interface

### **Business Success**

* [ ] Users can successfully create portfolio
* [ ] PDF export meets sharing requirements
* [ ] App handles multiple users concurrently
* [ ] Data integrity maintained
* [ ] User retention (measured by return visits)

This PRD provides comprehensive specifications for Cursor to implement CheckDisOut exactly as envisioned. Every technical detail, component specification, and implementation guideline is included for flawless execution.