

Frontend Development Guide - Captions AI Clone

Project Overview

वदअमतजमक जव जतनजप कसअ ०५५

Build a modern, responsive web application for AI-powered video editing and subtitle generation. The frontend will consume the backend APIs and provide an intuitive user experience similar to Captions AI.

Technology Stack

- **Framework:** React 18+ with TypeScript
- **Styling:** Tailwind CSS + Styled Components
- **State Management:** Redux Toolkit + RTK Query
- **Routing:** React Router v6
- **Video Player:** React Player + Video.js
- **File Upload:** React Dropzone
- **Real-time:** Socket.io Client
- **Animations:** Framer Motion
- **Icons:** Lucide React
- **Forms:** React Hook Form + Zod validation

Project Structure

```
frontend/  
├─ public/  
│   ├── index.html  
│   └─ favicon.ico
```

वदअमतजमक जव जतनजप कमअ ०५५

```
├─ components/  
│   ├── common/  
│   │   ├── Header/  
│   │   ├── Sidebar/  
│   │   ├── Modal/  
│   │   ├── Loading/  
│   │   └─ Toast/  
│   ├── auth/  
│   │   ├── LoginForm/  
│   │   ├── RegisterForm/  
│   │   └─ ForgotPassword/  
│   ├── dashboard/  
│   │   ├── ProjectCard/  
│   │   ├── StatsCard/  
│   │   └─ RecentProjects/  
│   ├── editor/  
│   │   ├── VideoPlayer/  
│   │   ├── Timeline/  
│   │   ├── ToolPanel/  
│   │   ├── SubtitleEditor/  
│   │   └─ ExportPanel/  
│   ├── ai/  
│   │   ├── AvatarCreator/  
│   │   ├── SubtitleGenerator/  
│   │   ├── AutoEditor/  
│   │   └─ ProcessingStatus/  
│   └─ upload/  
│       ├── FileUploader/  
│       ├── ProgressBar/  
│       └─ VideoPreview/  
├─ pages/  
│   ├── Landing/  
│   ├── Auth/  
│   ├── Dashboard/  
│   ├── Editor/  
│   ├── Projects/  
│   ├── Profile/  
│   └─ Pricing/  
├─ hooks/  
│   ├── useAuth.js  
│   ├── useSocket.js  
│   ├── useUpload.js  
│   ├── useVideo.js  
│   └─ useLocalStorage.js  
├─ store/  
│   ├── index.js  
│   ├── slices/  
│   │   └─ authSlice.js
```



Dependencies

json

```
{
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.15.0",
    "typescript": "^5.0.0",
    "@reduxjs/toolkit": "^1.9.5",
    "react-redux": "^8.1.2",
    "react-hook-form": "^7.45.4",
    "zod": "^3.22.2",
    "@hookform/resolvers": "^3.3.1",
    "tailwindcss": "^3.3.3",
    "styled-components": "^6.0.7",
    "framer-motion": "^10.16.4",
    "react-player": "^2.13.0",
    "video.js": "^8.5.2",
    "react-dropzone": "^14.2.3",
    "socket.io-client": "^4.7.2",
    "lucide-react": "^0.263.1",
    "react-hot-toast": "^2.4.1",
    "axios": "^1.5.0",
    "date-fns": "^2.30.0",
    "react-query": "^3.39.3",
    "wavesurfer.js": "^7.3.0"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.0.3",
    "vite": "^4.4.5",
    "eslint": "^8.45.0",
    "prettier": "^3.0.0",
    "@types/react": "^18.2.15",
    "@types/react-dom": "^18.2.7"
  }
}
```

वदअमतजमक जव जतनजप कमअ ०५५

Page Structure & Features

1. Landing Page (1)

Features:

- Hero section with demo video
- Feature highlights with animations
- Pricing plans
- User testimonials
- Call-to-action buttons

Components Needed:

- **HeroSection** - Main banner with video demo

- `FeatureGrid` - AI features showcase
- `PricingCards` - Subscription plans
- `TestimonialSlider` - User reviews

बदलते रहने वाले links और information कमअ ०५५

2. Authentication Pages (`/auth`)

Features:

- Login/Register forms
- Password reset functionality
- Form validation
- Loading states
- Error handling

Components Needed:

- `LoginForm` - Email/password login
- `RegisterForm` - User registration
- `ForgotPasswordForm` - Password reset
- `AuthWrapper` - Common layout
- `FormInput` - Reusable input component

3. Dashboard (`/dashboard`)

Features:

- Project overview cards
- Usage statistics
- Recent activity
- Quick actions
- Storage usage meter

Components Needed:

- `ProjectGrid` - All user projects
- `StatsOverview` - Usage metrics
- `QuickActions` - New project, upload video
- `RecentActivity` - Timeline of actions
- `UsageChart` - Visual usage data

4. Video Editor (`/editor/:projectId`)

Features:

- Video player with controls
- Timeline for editing

- Subtitle editor
- AI tools panel
- Real-time preview

• Export options

બ્લેકબોર્ડ પર જોઈ શકાય તેવું જોડાણ કરવા માટે ૦૫૫

Main Components:

- `VideoPlayer` - Main video display
- `Timeline` - Edit timeline
- `SubtitleEditor` - Caption editing
- `ToolPanel` - AI features
- `LayersPanel` - Video layers
- `ExportModal` - Export settings

5. Projects Page (`/projects`)

Features:

- Project management
- Search and filter
- Bulk operations
- Project templates
- Collaboration features

Components Needed:

- `ProjectList` - Grid/list view
- `SearchFilter` - Project filtering
- `ProjectCard` - Individual project
- `BulkActions` - Multiple project actions
- `TemplateGallery` - Project templates

Key Components Specifications

Video Player Component

typescript

```
interface VideoPlayerProps {  
  src: string;  
  subtitles?: Subtitle[];  
  onTimeUpdate: (time: number) => void;  
  onDurationChange: (duration: number) => void;  
  controls?: boolean;  
  autoPlay?: boolean;  
}
```

वदअमृतजमक जव जतनजप कमअ ०५५

Features:

- Custom video controls
- Subtitle overlay
- Playback speed control
- Fullscreen support
- Keyboard shortcuts
- Progress tracking

File Uploader Component

typescript

```
interface FileUploaderProps {  
  onUpload: (files: File[]) => void;  
  acceptedFormats: string[];  
  maxSize: number;  
  multiple?: boolean;  
  onProgress: (progress: number) => void;  
}
```

Features:

- Drag & drop interface
- Progress visualization
- File validation
- Preview thumbnails
- Upload to Cloudinary
- Error handling

Subtitle Editor Component

typescript

```
interface SubtitleEditorProps {  
  subtitles: Subtitle[];  
  currentTime: number;  
  onUpdate: (subtitles: Subtitle[]) => void;  
  onStyleChange: (style: SubtitleStyle) => void;  
}
```

Features:

- Real-time editing
- Time synchronization
- Style customization
- Language selection
- Auto-save functionality
- Export formats (SRT, VTT)

AI Tools Panel Component

typescript

```
interface AIToolsPanelProps {  
  videoId: string;  
  onProcessingStart: () => void;  
  onProcessingComplete: (result: any) => void;  
}
```

Features:

- Auto subtitle generation
- Smart video editing
- Avatar creation
- Scene detection
- Music suggestion
- Processing status

State Management Structure

Auth Slice

typescript

```
interface AuthState {  
  user: User | null;  
  token: string | null;  
  isAuthenticated: boolean;  
  loading: boolean;  
  error: string | null;  
}
```

Actions:

- login
- logout
- register
- updateProfile
- resetPassword

Project Slice

typescript

```
interface ProjectState {  
  projects: Project[];  
  currentProject: Project | null;  
  loading: boolean;  
  error: string | null;  
}
```

Actions:

- fetchProjects
- createProject
- updateProject
- deleteProject
- setCurrentProject

Video Slice

typescript

```
interface VideoState {  
  videos: Video[];  
  currentVideo: Video | null;  
  processing: boolean;  
  uploadProgress: number;  
  error: string | null;  
}
```

Actions:

- uploadVideo
- processVideo
- updateVideo
- deleteVideo
- setCurrentVideo

Editor Slice

typescript

```
interface EditorState {
  currentTime: number;
  duration: number;
  zoom: number;
};
subtitles: Subtitle[];
tools: {
  activeTools: string[];
  processing: boolean;
};
export: {
  settings: ExportSettings;
  progress: number;
};
}
```

Actions:

- updateTimeline
- addSubtitle
- updateSubtitle
- deleteSubtitle
- setExportSettings
- startExport

API Integration Patterns

RTK Query Setup

typescript

```
// Base API configuration
export const api = createApi({
  reducerPath: 'api',
  baseQuery: fetchBaseQuery({
    baseUrl: '/api',
    prepareHeaders: (headers, { getState }) => {
      const token = (getState() as RootState).auth.token;
      if (token) {
        headers.set('authorization', `Bearer ${token}`);
      }
      return headers;
    },
  }),
  tagTypes: ['User', 'Project', 'Video', 'Subtitle'],
  endpoints: (builder) => ({}),
});
```

API Endpoints Implementation

typescript

// Auth API

```
export const authApi = api.injectEndpoints({
  endpoints: (builder) => ({
    login: builder.mutation({
      query: (credentials) => ({
        url: '/auth/login',
        method: 'POST',
        body: credentials,
      }),
    }),
    // ... other auth endpoints
  }),
});
```

// Project API

```
export const projectApi = api.injectEndpoints({
  endpoints: (builder) => ({
    getProjects: builder.query({
      query: () => '/projects',
      providesTags: ['Project'],
    }),
    createProject: builder.mutation({
      query: (project) => ({
        url: '/projects',
        method: 'POST',
        body: project,
      }),
      invalidatesTags: ['Project'],
    }),
    // ... other project endpoints
  }),
});
```

Real-time Features

Socket.io Integration

typescript

// Socket hook

export const useSocket = () => {

const [socket, setSocket] = useState(null);

const { token } = useSelector((state) => state.auth);

useEffect(() => {

if (token) {

const newSocket = io(process.env.REACT_APP_API_URL, {
auth: { token },
});

setSocket(newSocket);

return () => newSocket.close();

}

}, [token]));

return socket;

};

// Processing status updates

useEffect(() => {

if (socket) {

socket.on('processing-update', (data) => {
dispatch(updateProcessingStatus(data));
});

socket.on('processing-complete', (data) => {
dispatch(processingComplete(data));
toast.success('Processing completed!');
});

}

}, [socket]);

वदअमृतजमक जव जतनजप कमअ ०५५

UI/UX Design Guidelines

Color Scheme

CSS

```
:root {
  --primary: #6366f1;      /* Indigo */
  --primary-dark: #4f46e5;
  --secondary: #8b5cf8;    /* Purple */
  --accent: #06b6d4;       /* Cyan */
  --success: #10b981;       /* Emerald */
  --warning: #f59e0b;       /* Amber */
  --error: #ef4444;         /* Red */
  --gray-50: #f9fafb;
  --gray-900: #111827;
  --background: #ffffff;
  --surface: #f8fafc;
}
```

वदअमृतजमक जव जतनजप कमअ ०५५

Typography

CSS

```
.text-heading-1 { @apply text-4xl font-bold leading-tight; }
.text-heading-2 { @apply text-3xl font-semibold leading-tight; }
.text-heading-3 { @apply text-2xl font-semibold leading-snug; }
.text-body-large { @apply text-lg leading-relaxed; }
.text-body { @apply text-base leading-normal; }
.text-body-small { @apply text-sm leading-normal; }
.text-caption { @apply text-xs leading-tight; }
```

Component Design Patterns

CSS

```
/* Card component */
.card {
  @apply bg-white rounded-lg shadow-sm border border-gray-200 p-6;
}

/* Button variants */
.btn-primary {
  @apply bg-primary text-white px-4 py-2 rounded-lg font-medium hover:bg-primary-dark transiti
}

.btn-secondary {
  @apply bg-gray-100 text-gray-700 px-4 py-2 rounded-lg font-medium hover:bg-gray-200 transiti
}

/* Input styling */
.input {
  @apply w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 foc
}
```

Performance Optimization

Code Splitting

typescript

// Lazy Load pages

व्यवस्थापक जवळून जपू करा ०५५
`const Dashboard = lazy(() => import('./pages/Dashboard'));
const Editor = lazy(() => import('./pages/Editor'));
const Projects = lazy(() => import('./pages/Projects'));`

// Route configuration

```
<Routes>  
  <Route path="/dashboard" element={  
    <Suspense fallback=<Loading />>  
      <Dashboard />  
    </Suspense>  
  } />  
</Routes>
```

Video Optimization

typescript

// Video player optimization

```
const VideoPlayer = ({ src, ...props }) => {  
  const [loading, setLoading] = useState(true);  
  const [error, setError] = useState(null);
```

```
  const videoOptions = {  
    fluid: true,  
    responsive: true,  
    preload: 'metadata',  
    controls: true,  
    sources: [{  
      src: src,  
      type: 'video/mp4'  
    }]  
  };
```

```
  return (  
    <div className="video-container">  
      {loading && <VideoLoader />}  
      <ReactPlayer  
        url={src}  
        controls  
        width="100%"  
        height="100%"  
        onReady={() => setLoading(false)}  
        onError={(e) => setError(e)}  
        {...props}  
      />  
    </div>  
  );  
};
```

Memory Management

typescript

// Cleanup effects

वदअमतजमक> जव जतनजप कमअ ०५५

```
useEffect(() => {
  const handleResize = () => {
    // Handle window resize
  };

  window.addEventListener('resize', handleResize);

  return () => {
    window.removeEventListener('resize', handleResize);
  };
}, []);

// Video cleanup
useEffect(() => {
  return () => {
    if (videoRef.current) {
      videoRef.current.pause();
      videoRef.current.src = '';
      videoRef.current.load();
    }
  };
}, []);
```

Testing Strategy

Component Testing

typescript

// Example test for VideoPlayer

```
import { render, screen } from '@testing-library/react';
import VideoPlayer from './VideoPlayer';

test('renders video player with controls', () => {
  render(<VideoPlayer src="test-video.mp4" controls />);
  expect(screen.getByRole('video')).toBeInTheDocument();
});

test('displays loading state initially', () => {
  render(<VideoPlayer src="test-video.mp4" />);
  expect(screen.getByText('Loading...')).toBeInTheDocument();
});
```

Integration Testing

typescript

```
// API integration tests
test('uploads video successfully', async () => {
  const file = new File(['video content'], 'test.mp4', { type: 'video/mp4' });
  const mockUpload = jest.fn().mockResolvedValue({ success: true });

  render(<FileUploader onUpload={mockUpload} />);

  // Simulate file drop
  fireEvent.drop(screen.getByText('Drop files here'), {
    dataTransfer: { files: [file] }
  });

  await waitFor(() => {
    expect(mockUpload).toHaveBeenCalledWith([file]);
  });
});
```

वदअमृतजमक जव जतनजप कमअ ०५५

Deployment Configuration

Environment Variables

env

```
REACT_APP_API_URL=http://localhost:5000/api
REACT_APP_SOCKET_URL=http://localhost:5000
REACT_APP_CLOUDINARY_CLOUD_NAME=your_cloud_name
REACT_APP_CLOUDINARY_UPLOAD_PRESET=your_preset
REACT_APP_ENVIRONMENT=development
```

Build Configuration

json

```
{
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "preview": "vite preview",
    "test": "jest",
    "lint": "eslint src --ext ts,tsx",
    "format": "prettier --write src"
  }
}
```

Setup Commands

Project Initialization

bash

Create React app with Vite

npm create vite@latest captions-frontend -- --template react-ts

वदअमृतजमक जव जतनजप कमअ ०५५

Install dependencies

npm install react-router-dom @reduxjs/toolkit react-redux react-hook-form zod @hookform/resolve

Install dev dependencies

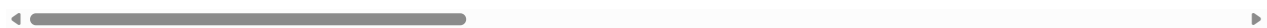
npm install -D @vitejs/plugin-react vite eslint prettier @types/react @types/react-dom

Initialize Tailwind CSS

npx tailwindcss init -p

Create project structure

mkdir -p src/{components/{common,auth,dashboard,editor,ai,upload},pages,hooks,store/{slices,api



Development Workflow

1. **Setup:** Run initialization commands
2. **Development:** Use `npm run dev` for hot reloading
3. **Testing:** Run `npm test` for component tests
4. **Building:** Use `npm run build` for production
5. **Linting:** Run `npm run lint` for code quality

Key Implementation Notes

1. **Responsive Design:** Mobile-first approach with Tailwind CSS
2. **Accessibility:** WCAG 2.1 AA compliance
3. **Performance:** Code splitting and lazy loading
4. **User Experience:** Smooth animations and transitions
5. **Error Handling:** Comprehensive error boundaries
6. **Real-time Updates:** Socket.io for live processing status
7. **Offline Support:** Service worker for offline functionality
8. **SEO:** Meta tags and structured data
9. **Analytics:** User interaction tracking
10. **Security:** XSS protection and secure API calls

This guide provides everything needed to build a modern, scalable frontend for your Captions AI clone.