Motion Tracker

Phase 1: Detailed Technical Specification

Implementation-Ready PRD for Core Features

Version 1.0 | January 2026

1. Overview

This document provides implementation-level specifications for Phase 1 of Motion Tracker. It covers the six core modules that form the foundation of video-based motion analysis, derived from common requirements across AP Physics 1 laboratory investigations.

1.1 Core Modules

1. **Video Engine:** Upload, playback, frame extraction, navigation
2. **Coordinate System:** Scale calibration, origin placement, axis rotation
3. **Point Tracking:** Manual marking, path visualization, undo/redo
4. **Data Table:** Time series display, calculated columns, editing
5. **Graphing:** Multiple plot types, regression, graph-video sync
6. **Project Management:** Cloud storage, auth, export

1.2 Tech Stack

Layer	Technology
Framework	React 18 + TypeScript + Vite
Styling	Tailwind CSS + shadcn/ui
State Management	Zustand with immer middleware
Charts	Recharts
Backend	Supabase (Auth, PostgreSQL, Storage)
Hosting	Railway

2. Module 1: Video Engine

The video engine handles all aspects of video input, playback, and frame extraction. This is the foundation upon which all analysis is built.

2.1 Functional Requirements

2.1.1 Video Upload

- Accept video files via drag-and-drop or file picker
- Supported formats: MP4 (H.264), WebM (VP8/VP9), MOV (with warning for non-Safari)
- Maximum file size: 500MB (configurable)
- Extract and display video metadata: duration, frame rate, resolution
- Generate thumbnail on upload for project list
- Upload to Supabase Storage with progress indicator

2.1.2 Video Playback

- Play/pause toggle (spacebar shortcut)
- Playback speed: 0.25x, 0.5x, 1x, 2x
- Timeline scrubber with frame-accurate seeking
- Current time display in seconds and frame number

2.1.3 Frame Navigation

- Step forward one frame (right arrow or '.' key)
- Step backward one frame (left arrow or ',' key)
- Jump forward 10 frames (Shift + right arrow)
- Jump backward 10 frames (Shift + left arrow)
- Go to first frame (Home key)
- Go to last frame (End key)
- Frame input field for direct navigation

2.1.4 Frame Extraction

- Render current frame to Canvas element for overlay drawing
- Maintain aspect ratio with letterboxing/pillarboxing
- Support pinch-to-zoom and pan for precise analysis

2.2 Data Model

VideoMetadata type stored with each project:

Field	Type	Description
storageUrl	string	Supabase Storage URL
fileName	string	Original file name
duration	number	Duration in seconds
frameRate	number	Frames per second (detected or user-specified)
width	number	Video width in pixels
height	number	Video height in pixels
totalFrames	number	Calculated: Math.floor(duration * frameRate)
thumbnailUrl	string	URL to generated thumbnail

2.3 Key Implementation Notes

Frame Rate Detection: HTML5 video doesn't expose frame rate directly. Use requestVideoFrameCallback() where supported (Chrome, Edge) to measure actual frame intervals. Fall back to common rates (30, 60, 24) with user override option.

Seeking Precision: Use video.currentTime = frameNumber / frameRate for frame-accurate seeking. After setting, wait for 'seeked' event before drawing to canvas.

Canvas Rendering: On each frame change, draw video to canvas with ctx.drawImage(video, 0, 0). Canvas must be sized to match video dimensions for accurate coordinate mapping.

2.4 Acceptance Criteria

- User can upload MP4 video up to 500MB
- Video metadata is extracted and displayed correctly
- Frame-by-frame navigation is accurate within 1 frame
- All keyboard shortcuts work as specified
- Video uploads to Supabase Storage with progress feedback

3. Module 2: Coordinate System

The coordinate system module establishes the mapping between pixel coordinates in the video and real-world physical units. This is critical for all quantitative analysis.

3.1 Functional Requirements

3.1.1 Scale Calibration

Students must establish a scale by marking two points of known distance in the video. This is typically done using a meter stick or other reference object visible in the frame.

- User clicks two points on the video to define a reference distance
- User enters the real-world distance between the points
- User selects unit: meters (m), centimeters (cm), millimeters (mm), feet (ft), inches (in)
- Display calculated scale factor (pixels per unit)
- Allow re-calibration at any time
- Visual indicator showing scale line on video

3.1.2 Origin Placement

The origin defines the (0, 0) point for all position measurements. Placement depends on the experiment type.

- User clicks to place origin point
- Origin displayed with crosshairs and X/Y axis lines
- Draggable for repositioning
- Default: bottom-left of video frame

3.1.3 Axis Orientation

For inclined plane experiments (Investigation 1, 4), students need to align the coordinate system with the ramp surface.

- Rotation handle on X-axis for tilting coordinate system
- Angle input field for precise rotation (degrees)
- Snap to common angles: 0°, 30°, 45°, 60°, 90°
- Display current rotation angle
- Y-axis always perpendicular to X-axis

3.1.4 Y-Axis Direction

- Toggle: positive Y up (physics convention) or positive Y down (video convention)
- Default: positive Y up

3.2 Data Model

Field	Type	Description
scalePoint1	{ x: number, y: number }	First scale reference point (pixels)
scalePoint2	{ x: number, y: number }	Second scale reference point (pixels)
scaleDistance	number	Real-world distance value
scaleUnit	'm'|'cm'|'mm'|'ft'|'in'	Unit of measurement
origin	{ x: number, y: number }	Origin position (pixels)
rotation	number	Rotation angle in degrees (0-360)
yAxisUp	boolean	true = positive Y is up

3.3 Coordinate Transformation

The core calculation that converts pixel coordinates to world coordinates:

Step 1 - Calculate scale factor: pixelDistance = sqrt((p2.x - p1.x)² + (p2.y - p1.y)²), then pixelsPerUnit = pixelDistance / scaleDistance

Step 2 - Translate to origin: translated = { x: pixel.x - origin.x, y: pixel.y - origin.y }

Step 3 - Apply rotation: rotated.x = translated.x * cos(θ) + translated.y * sin(θ), rotated.y = -translated.x * sin(θ) + translated.y * cos(θ)

Step 4 - Apply Y-axis direction: if (yAxisUp) rotated.y = -rotated.y

Step 5 - Convert to world units: world = { x: rotated.x / pixelsPerUnit, y: rotated.y / pixelsPerUnit }

3.4 Acceptance Criteria

- User can set scale using two points and a known distance
- Origin can be placed and repositioned by dragging
- Axes can be rotated to any angle
- Coordinate system overlays remain visible during tracking
- Calculated world coordinates are accurate to 3 decimal places

4. Module 3: Point Tracking

Point tracking is the core interaction where students mark the position of an object in each video frame. This module handles input, visualization, and history management.

4.1 Functional Requirements

4.1.1 Manual Point Marking

- Click/tap on video canvas to mark object position
- Auto-advance to next frame after marking (configurable)
- Visual feedback: point appears with timestamp label
- Zoom mode for precise placement (2x, 4x magnification centered on cursor)
- Skip frames option: mark every Nth frame (useful for fast motion)

4.1.2 Point Adjustment

- Click existing point to select it
- Drag selected point to new position
- Delete selected point (Delete or Backspace key)
- Click on data table row to select corresponding point

4.1.3 Path Visualization

- Draw path connecting all tracked points
- Trail length: show all points, last N points, or current frame only
- Point size and color customization
- Highlight current frame's point
- Toggle visibility of path overlay

4.1.4 Undo/Redo

- Undo last action (Ctrl/Cmd + Z)
- Redo undone action (Ctrl/Cmd + Shift + Z or Ctrl/Cmd + Y)
- History depth: 50 actions minimum
- Actions tracked: add point, move point, delete point, change coordinate system

4.2 Data Model

DataPoint type for each tracked measurement:

Field	Type	Description
id	string	Unique identifier (UUID)

frameNumber	number	Frame index (0-based)
time	number	Time in seconds (frameNumber / frameRate)
pixelX	number	X position in pixels
pixelY	number	Y position in pixels

Note: worldX and worldY are calculated on-the-fly from pixel coordinates using the current coordinate system settings. This allows recalculation if the user adjusts the coordinate system after tracking.

4.3 Acceptance Criteria

- User can mark points by clicking on video
- Points can be selected, moved, and deleted
- Path visualization updates in real-time
- Undo/redo works for all point operations
- Auto-advance after marking is toggleable

5. Module 4: Data Table

The data table displays all tracked measurements and calculated values. It's the primary interface for viewing quantitative data and is essential for AP Physics lab reports.

5.1 Functional Requirements

5.1.1 Base Columns (Always Present)

Column	Header	Description
Row #	#	Sequential index (1, 2, 3...)
Time	t (s)	Time in seconds from frame number
X Position	x (m)	World X coordinate (unit from calibration)
Y Position	y (m)	World Y coordinate (unit from calibration)

5.1.2 Calculated Columns

These columns are computed using central difference method where possible for better accuracy:

Column	Header	Calculation
X Velocity	vx (m/s)	(x[n+1] - x[n-1]) / (t[n+1] - t[n-1])
Y Velocity	vy (m/s)	(y[n+1] - y[n-1]) / (t[n+1] - t[n-1])
Speed	|v| (m/s)	sqrt(vx^2 + vy^2)
X Acceleration	ax (m/s^2)	(vx[n+1] - vx[n-1]) / (t[n+1] - t[n-1])
Y Acceleration	ay (m/s^2)	(vy[n+1] - vy[n-1]) / (t[n+1] - t[n-1])

5.1.3 Derived Columns (User-Created)

For linearization and custom analysis, users can add derived columns:

- t^2 — Time squared (for verifying constant acceleration: x vs t^2 is linear)
- x^2 — Position squared (for energy relationships)
- 1/t — Inverse time
- sqrt(x) — Square root of position
- Custom formula using column references (e.g., "x * 2 + y")

5.1.4 Table Interactions

- Click row to select and highlight corresponding point on video
- Double-click cell to edit pixel coordinates (recalculates world values)
- Sort by any column (ascending/descending)
- Toggle column visibility
- Resize columns by dragging
- Scroll synced with video frame when playing

5.2 Acceptance Criteria

- Table displays all tracked points with time and position
- Velocity and acceleration columns calculate correctly
- User can add derived columns (t^2, x^2, etc.)
- Row selection syncs with video and graph highlighting
- Table updates immediately when coordinate system changes

6. Module 5: Graphing

The graphing module is critical for physics analysis. Students use graphs to identify motion types, verify relationships, and extract physical quantities like slope (velocity from x-t graphs, acceleration from v-t graphs).

6.1 Functional Requirements

6.1.1 Graph Types

The following graph configurations must be supported to cover all AP Physics 1 lab requirements:

Graph	Purpose	Slope Meaning
x vs t	Horizontal position over time	Horizontal velocity (vx)
y vs t	Vertical position over time	Vertical velocity (vy)
vx vs t	Velocity over time	Acceleration (ax)
vy vs t	Vertical velocity over time	Vertical acceleration (ay ≈ g)
y vs x	Trajectory shape	Path visualization
x vs t^2	Linearize constant acceleration	½ acceleration (½a)
Custom	Any column vs any column	Depends on variables

6.1.2 Linear Regression

Linear regression is essential for analyzing motion and extracting physical quantities. For every graph:

- Toggle best-fit line on/off
- Display equation: y = mx + b
- Display R^2 correlation coefficient
- Display slope (m) with units
- Display y-intercept (b) with units
- Option to fit only selected data range

6.1.3 Graph Interactions

- Click data point to highlight and jump to corresponding video frame
- Hover to show tooltip with exact values
- Pinch/scroll to zoom

- Drag to pan when zoomed
- Double-click to reset zoom
- Current video frame position shown as vertical line on time-based graphs

6.1.4 Graph Panel Layout

- Single graph view (maximized)
- Dual graph view (stacked vertically, for comparing x-t and v-t)
- Resizable panel (drag divider between video and graphs)
- Collapse/expand graph panel

6.2 Acceptance Criteria

- All standard graph types (x-t, y-t, v-t, y-x) can be displayed
- Custom axis selection works with any column
- Linear regression displays correct slope, intercept, and R^2
- Clicking graph point jumps video to that frame
- Graphs update in real-time as points are added

7. Module 6: Project Management

Project management handles authentication, cloud storage, and data export. This ensures student work is saved and accessible across devices.

7.1 Functional Requirements

7.1.1 Authentication

- Email/password registration and login
- Google OAuth ("Sign in with Google")
- Password reset via email
- Guest mode: use without account (local storage only, prompt to create account to save)
- Session persistence (stay logged in)

7.1.2 Project CRUD

- Create new project with name
- List all user's projects with thumbnails, names, last modified date
- Open existing project
- Rename project
- Duplicate project
- Delete project (with confirmation)
- Auto-save every 30 seconds and on significant actions

7.1.3 Sharing

- Generate shareable link (view-only access)
- Shared projects are read-only for viewers
- Viewers can duplicate to their own account to edit
- Revoke sharing access

7.1.4 Export

- CSV export of data table (all columns or selected columns)
- PNG export of each graph
- PNG export of video frame with overlay
- Copy data to clipboard

7.2 Database Schema

Supabase PostgreSQL tables:

users: Managed by Supabase Auth (id, email, created_at)

projects: id (uuid), user_id (fk), name, created_at, updated_at, is_public, share_token

project_data: id (uuid), project_id (fk), video_metadata (jsonb), coordinate_system (jsonb), data_points (jsonb), settings (jsonb)

7.3 Acceptance Criteria

- Users can register, login, and logout
- Projects persist across sessions and devices
- Auto-save prevents data loss
- CSV export produces valid, importable files
- Share links work for non-authenticated viewers

8. UI Layout Specification

The application uses a responsive layout optimized for desktop and tablet use during lab sessions.

8.1 Main Analysis View

The primary workspace is divided into three resizable panels:

Left Panel - Video (60% default width): Video canvas with overlay, playback controls below, frame navigation, timeline scrubber

Right Panel - Split Top/Bottom: Top: Graph panel (selectable graph type, regression toggle). Bottom: Data table (scrollable, column toggles)

Toolbar (Top): Project name, save status, mode selector (Setup/Track/Analyze), settings menu, export menu, user menu

8.2 Mode-Based UI

The interface adapts based on current task:

Setup Mode: Shows coordinate system tools prominently (scale, origin, axis rotation). Video overlay shows coordinate grid. Tracking and graphs are visible but de-emphasized.

Track Mode: Point marking is primary interaction. Auto-advance toggle visible. Trail visualization controls. Data table shows live updates.

Analyze Mode: Graph panel maximized. Regression controls prominent. Data table fully interactive. Video serves as reference.

8.3 Responsive Breakpoints

- Desktop (≥1280px): Full three-panel layout
- Tablet (768px-1279px): Stacked layout, video on top, graph/table tabbed below
- Mobile (<768px): Warning that desktop/tablet is recommended; simplified single-panel view

9. Development Schedule

Estimated timeline for Phase 1 implementation, assuming part-time development:

Week	Module	Key Deliverables
1-2	Project Setup + Auth	Repo, Supabase config, login/register
3-4	Video Engine	Upload, playback, frame navigation
5-6	Coordinate System	Scale, origin, axis rotation
7-8	Point Tracking	Marking, path viz, undo/redo
9-10	Data Table	Columns, calculations, derived cols
11-12	Graphing	All graph types, regression, sync
13-14	Project Management + Export	Save/load, sharing, CSV/PNG export
15-16	Polish + Deploy	UI refinement, Railway deploy, beta

10. Success Criteria for Phase 1

Phase 1 is complete when a user can:

1. Create an account and log in
2. Upload a video of a ball rolling down a ramp
3. Set scale using a meter stick visible in the video
4. Place origin at the bottom of the ramp
5. Rotate axes to align with the ramp surface
6. Track the ball position frame-by-frame
7. View position-time and velocity-time graphs
8. Add a best-fit line and read the slope (acceleration)
9. Create a linearized plot (x vs t^2) to verify constant acceleration
10. Export data as CSV for use in a lab report
11. Export graph as PNG for inclusion in a lab report
12. Save the project and reopen it later
13. Share the project with a teacher via link

This workflow mirrors AP Physics 1 Investigation 1 (1D and 2D Kinematics) and validates that the tool is ready for classroom use.