Developer: Tiffany Morgan
Project Name: Launch Commander
Course: INEW 2334
Date: January 28, 2026
Project Type: Full-stack MEAN Application

Proposal Document: Launch Commander

Executive Summary

Launch Commander is an interactive web application that bridges the gap between aerospace data and public engagement by transforming real-world rocket launch information into an engaging, gamified prediction platform. Users can track upcoming launches from major providers (Firefly, SpaceX, NASA, etc.) and predict mission outcomes, competing on a dynamic leaderboard while learning about aerospace operations.

Project Description

Launch Commander is a full-stack MEAN (MongoDB, Express.js, Angular, Node.js) application that integrates with the Launch Library 2 API to provide real-time rocket launch tracking and prediction functionality. The platform allows users to make informed predictions about mission parameters including launch success, engine performance metrics, orbital achievements, and payload deployment outcomes. An automated scoring system evaluates predictions against actual telemetry data, fostering competitive engagement through leaderboard rankings.

Problem Statement

The aerospace industry generates vast amounts of publicly available launch data, yet meaningful public engagement with this information remains limited. Existing launch tracking websites provide passive viewing experiences without interactive elements that encourage deeper understanding of mission parameters or sustained user engagement. Additionally, there is a disconnect between technical aerospace operations and public comprehension of launch complexity.

Proposed Solution

Launch Commander addresses these gaps by:

- Democratizing Aerospace Data: Making complex launch information accessible and engaging for general audiences
- Interactive Learning: Users gain familiarity with mission-critical parameters (MECO timing, orbital mechanics, payload operations) through active prediction

- Community Engagement: Competitive leaderboard system creates sustained interest in aerospace events
- Real-Time Integration: Automated data synchronization ensures current information from authoritative sources

Target Audience

Primary Users

Aerospace Enthusiasts: Individuals interested in space exploration and rocket technology
STEM Students: Learners seeking practical application of physics and engineering concepts
Industry Professionals: Aerospace workers looking for casual engagement with launch schedules

Client Needs Addressed

1. Engagement: Transform passive data consumption into active participation
2. Education: Provide hands-on learning about mission-critical aerospace parameters
3. Community: Create competitive environment fostering sustained interest
4. Accessibility: Make complex technical data understandable for general audiences
5. Real-Time Updates: Ensure current information from authoritative sources

Secondary Stakeholders

Educational Institutions: Potential classroom tool for physics/engineering curriculum
Aerospace Companies: Demonstration of public engagement strategies for technical content
Developer Community: Example of MEAN stack implementation with external API integration

Key Features

Core Functionality

1. Real-Time Launch Tracking
   - Live data synchronization with Launch Library 2 API
   - Display of upcoming launches with vehicle, site, and mission details
   - Automatic status updates as launch dates approach

2. Prediction System
   - Multi-parameter prediction forms (success, timing, altitude, payload)
   - Validation to prevent duplicate predictions
   - Timestamp tracking for prediction history

3. Automated Scoring Engine
   - Algorithm evaluates predictions against actual outcomes
   - Weighted scoring based on prediction accuracy
   - Point accumulation for leaderboard ranking

4. Dynamic Leaderboard

   - Real-time ranking updates as launches complete
   - User statistics including total predictions and accuracy rates
   - Competitive display encouraging continued engagement

5. Responsive Interface
   - Angular Material Design components
   - Three-column layout optimized for desktop viewing
   - Mobile-responsive design for on-the-go access

Technical Approach

Architecture
- Frontend: Angular framework with TypeScript, Angular Material UI components
- Backend: Express.js REST API with middleware for CORS and JSON parsing
- Database: MongoDB for persistent storage of launches, predictions, and leaderboard data
- External Integration: HTTPS requests to Launch Library 2 API with streaming data handling
- Automation: Background task scheduling for launch status monitoring

Data Flow
1. Backend fetches launch data from external API on server startup
2. Frontend requests data via REST endpoints
3. Users submit predictions through Angular forms
4. Backend validates and stores predictions in MongoDB
5. Automated scoring runs when launch status updates to "completed"
6. Leaderboard recalculates rankings and updates frontend display

Key Technical Implementations
- HTTP Streaming: Efficient handling of large API responses via chunk processing
- Asynchronous Operations: Promise-based database queries and API calls
- Two-Way Data Binding: Angular's [(ngModel)] for reactive form inputs
- RESTful Design: Clear endpoint structure for resource management
- Data Validation: Server-side checks for prediction integrity

Front-End Design
- Framework: Angular 14+ with TypeScript for type-safe component development
- UI Library: Angular Material Design system for consistent, modern interface
- Layout: Responsive 3-column grid (launches, predictions, leaderboard)
- State Management: Component-based state with RxJS observables for data streams
- Forms: Template-driven forms with ngModel for real-time validation
- Styling: Custom CSS with Material theming for aerospace-inspired design
- User Feedback: Toast notifications for success/error states, loading indicators

Back-End Design
- Server Framework: Express.js with middleware pipeline architecture

- Database: MongoDB with three collections (launches, predictions, leaderboard)
- API Endpoints: 5 RESTful routes (GET launches, GET/POST predictions, GET leaderboard, PUT admin updates)
- Data Processing: Automated scoring algorithm triggered by launch status changes
- Background Tasks: Node.js setInterval for periodic API synchronization
- Error Handling: Try-catch blocks with detailed logging for debugging

Security Measures
- CORS Configuration: Whitelist frontend origin to prevent unauthorized API access
- Input Validation: Server-side sanitization of user-submitted prediction data
- MongoDB Injection Prevention: Parameterized queries via MongoDB driver methods
- API Rate Limiting: Throttling mechanisms for external API calls to prevent abuse
- Environment Variables: Sensitive configuration stored outside codebase (.env files)
- Data Type Validation: TypeScript interfaces enforce strict data contracts
- Duplicate Prevention: Database constraints prevent multiple predictions per user/launch
- Error Exposure Limitation: Generic error messages to frontend, detailed logs server-side

Future Security Enhancements:
- User authentication via JWT tokens or OAuth
- Password hashing with bcrypt
- HTTPS enforcement for production deployment
- API key rotation policies
- SQL injection protection (when scaling to relational databases)

Hosting and Deployment
Development Environment:
- Local MongoDB instance on port 27017
- Express backend on localhost:3000
- Angular dev server on localhost:4200

Production Deployment Strategy:

Option 1: Cloud-Based (Recommended)
- Frontend Hosting: Vercel or Netlify for Angular static build
  - Automatic CI/CD from Git repository
  - Global CDN distribution for fast load times
  - Free SSL certificates
- Backend Hosting: Heroku, Railway, or DigitalOcean App Platform
  - Node.js runtime environment
  - Automatic scaling capabilities
  - Environment variable management
- Database Hosting: MongoDB Atlas (Cloud DBaaS)
  - Free tier supports up to 512MB storage
  - Automated backups and redundancy

  - Geographic distribution options

Option 2: Traditional VPS
- Provider: DigitalOcean Droplet or AWS EC2
- Configuration: Ubuntu Linux server with Nginx reverse proxy
- Process Management: PM2 for Node.js application monitoring
- Database: Self-hosted MongoDB or managed service

Hosting Requirements:
- Backend: Minimum 512MB RAM, Node.js 16+
- Database: 1GB storage for initial deployment, scalable to 10GB+
- Frontend: Static file hosting (minimal resources)
- Bandwidth: Estimated 50GB/month for moderate traffic

Deployment Workflow:
1. Build Angular frontend for production (`ng build --prod`)
2. Push backend code to hosting platform
3. Configure environment variables (MongoDB URI, API keys)
4. Set up MongoDB Atlas cluster and whitelist server IP
5. Deploy frontend static files to CDN
6. Configure custom domain and SSL certificate
7. Implement monitoring (uptime checks, error tracking)

Project Objectives

Short-Term Goals
- Implement complete CRUD operations for launches and predictions
- Integrate external aerospace API for real-world data
- Deploy functional prediction and scoring system
- Create responsive, professional UI with Angular Material

Long-Term Vision
- Enhanced Analytics: User performance graphs and prediction trend analysis
- Social Features: User profiles, prediction sharing, achievement badges
- Expanded Data Sources: Integration with additional aerospace APIs for telemetry details
- Mobile Application: Native iOS/Android apps for broader accessibility
- Admin Dashboard: Content management system for manual launch updates
- Notification System: Email/push notifications for upcoming launches users have predicted

Intended Direction

Immediate Development Path
Launch Commander currently serves as a demonstration of full-stack MEAN development capabilities with emphasis on:

- External API integration patterns
- Real-time data synchronization
- Automated business logic (scoring algorithms)
- Professional UI/UX design principles

Future Enhancements

1. *Advanced Prediction Parameters*
   - Weather impact predictions
   - Specific payload deployment windows
   - Recovery success predictions for reusable boosters

2. *Community Features*
   - Discussion forums for launch events
   - Expert commentary integration
   - User-generated content (launch photos, livestream links)

3. *Educational Content*
   - Glossary of aerospace terminology
   - Mission parameter explanations
   - Historical launch data comparisons

4. *Gamification Expansion*
   - Achievement system for prediction milestones
   - Seasonal leaderboard competitions
   - Team-based prediction leagues

Expected Outcomes

Technical Demonstration

- Proof of competency in full-stack JavaScript development
- Understanding of asynchronous programming patterns
- Experience with NoSQL database design and operations
- Practical application of RESTful API principles

Portfolio Impact

- Showcase project relevant to aerospace industry interests
- Demonstrate ability to integrate complex external data sources
- Highlight problem-solving approach to user engagement challenges
- Evidence of professional-grade UI design sensibilities

Potential Industry Application

Launch Commander's architecture and engagement model could inform:
- Public outreach strategies for aerospace companies
- Educational tools for STEM programs

- Internal applications for launch operations teams
- Fan engagement platforms for space industry marketing

Success Metrics

Functional Requirements

- Real-time launch data retrieval and display
- User prediction submission and storage
- Automated scoring algorithm execution
- Dynamic leaderboard ranking system
- Responsive multi-column layout

Technical Excellence

- Clean, maintainable code with comprehensive commenting
- Efficient database queries with appropriate indexing
- Error handling for API failures and database connectivity
- Type safety through TypeScript implementation
- Modular component architecture

User Experience

- Intuitive prediction form workflows
- Clear visual hierarchy for information display
- Responsive feedback for user actions (success/error messages)
- Smooth animations and transitions
- Accessible design principles

Conclusion

Launch Commander represents the intersection of aerospace industry data and public engagement technology. By transforming passive launch tracking into an interactive prediction game, the application demonstrates both technical full-stack development capabilities and creative problem-solving for user engagement challenges. The project establishes a foundation for expanded aerospace-focused web applications while serving as a portfolio piece showcasing MEAN stack proficiency and external API integration expertise.