**Task Generator Project Documentation**

*Author: Lilla Pletser  
Date: April 19, 2025*

1. **Overview**

I developed the Task Generator as a lightweight, browser-based web application using F# and modern web technologies. The app delivers three daily challenges at Easy, Normal, and Hard levels, allowing users to mark tasks as completed or failed. A built‑in calendar view tracks performance history, and the UI supports both English and Hungarian.

1. **Technology Stack**

* Frontend: HTML5, CSS3, Bootstrap 5, JavaScript
* Language: F# (for client logic) compiled to JavaScript
* UI Effects: Canvas-based starfield animation
* State Management: LocalStorage for per‑day attempt count and history
* Localization: Simple dictionary in JS for EN/HU

1. **Development Process**

* Initial Setup:
* Created an index.html skeleton with Bootstrap and Google Fonts.
* Set up a <canvas> element for the starfield background.
* Core Logic in JS/F#:
* Defined task arrays and localization dictionaries.
* Implemented attempt tracking (MAX = 3) and daily reset at 23:59:59 via setInterval.
* Built handlers for Generate, Complete, and Fail buttons that update both UI and localStorage.
* Calendar Component:
* Designed a month‑based calendar render function.
* Added Previous/Next month navigation to view past/future history.
* Highlighted days in green/red based on success/failure from stored history.
* UI Polish:
* Added starfield animation using Canvas and requestAnimationFrame.
* Styled buttons, panels, and countdown with CSS to achieve a modern, dark theme.
* Ensured responsiveness across devices using Bootstrap grid.
* Localization:
* Stored current language in localStorage.
* Toggles EN/HU labels and text dynamically, without page reload.
* Testing & Refinement:
* Verified daily reset logic around midnight edge cases.
* Tested calendar navigation for month boundaries (year rollover).
* Ensured correct handling of button states when limits are reached.

1. **Deployment**

* Local Preview: Served via http-server or by opening index.html directly.
* Production (Azure):
* Manually uploaded the index.html and related assets to Azure Static Web Apps through the Azure Portal.
* No automated CI/CD pipeline is configured; updates are performed by republishing the site manually.

1. **Next Steps**

* Add server‑side leaderboard and data persistence.
* Introduce user accounts and remote sync.
* Expand task categories and analytics dashboard.

**----------------------------Future development expected----------------------------**