DEPLOYMENT PLAN

Cevdet Onat Cerit – 231101078
Alvin Dora Akıncı – 231101052
Emin Arslan – 231101007
Nehir Tıraş – 231101065
Nehir Aydın – 231101053

Task Matrix

Section	Responsible
Deployment Overview	Cevdet Onat Cerit
Deployment Process	Alvin Dora Akıncı
Configuration Plan	Emin Arslan & Nehir Aydın
Documentation Integration	Nehir Tıraş

Table of Contents

1. Introduction

This Deployment Plan describes how the StudyPilot web application was prepared, configured, and deployed for the PA3 project demonstration. StudyPilot is a fully client-side productivity platform designed for students, featuring task management, a Pomodoro timer, weather-based study recommendations, analytics, and Spotify music integration.

Because the system operates entirely in the browser and requires no backend server, deployment was carried out using a static hosting platform. This document outlines the selected approach, deployment procedures, configuration decisions, and the environment used to run the final version of the system.

---

2. Deployment Overview

2.1 Deployment Strategy

The StudyPilot system was deployed using a **static hosting strategy**, as all functionality is implemented using HTML, CSS, and JavaScript modules running directly on the client side. No server-side logic or backend database is required. This makes Netlify the most efficient option due to its simplicity, instant updates, and automatic HTTPS support.

The application interacts with external services—Spotify Web API and OpenWeatherMap API—using REST calls. All persistent data is managed through LocalStorage, making the system entirely self-contained.

2.2 Tools and Technologies

- **Netlify** – Static hosting & automatic deployment
- **GitHub** – Version control and project sharing
- **Visual Studio Code** – Code editor
- **Google Chrome DevTools** – Performance evaluation and testing
- **Spotify Web API (OAuth 2.0 PKCE)** – Focus music playlists
- **OpenWeatherMap API** – Weather data and study suggestions

---

2.3 Hosting Environment

StudyPilot was deployed on **Netlify** using the manual deployment method (drag-and-drop). Netlify automatically generates a public subdomain and delivers the static files over HTTPS.

The deployment environment includes:

- Static file hosting
- Automated CDN distribution
- Built-in SSL/TLS
- No server configuration required

2.4 Live Deployment URL

The final deployed version is accessible at:

https://cool-zuccutto-dd7c3c.netlify.app/index.html

This is the URL used for the PA3 presentation.

---

3. Deployment Process

3.1 Pre-Deployment Preparation

Before deployment, the project was restructured to ensure clean separation of UI, logic, and API responsibilities. All JavaScript modules were verified to load correctly through the main index.html entry point, and API integrations were tested locally to ensure HTTPS compatibility.

Unused or development-only files were removed to ensure a lightweight deployment package.

3.2 Project Structure

The final directory structure prepared for deployment is:

```
STUDYPILOT/
│
├── components/                 # UI components
│   ├── DashboardUI.js
│   ├── TaskCard.js
│   └── TimerUI.js
│
├── modules/                    # Core application logic
│   ├── Analytics.js
│   ├── Pomodoro.js
│   ├── Prioritizer.js
│   └── TaskManager.js
│
├── services/                   # External API handlers
│   ├── SpotifyService.js
│   └── WeatherService.js
│
├── effects/                    # Visual effects
│   └── starfield.js
│
├── styles/                     # CSS styling
│   ├── login.css
│   └── style.css
│
├── index.html                  # Main dashboard page
├── login.html                  # Login screen
└── login.js                    # Login workflow logic
```

This structure supports modularity, maintainability, and compatibility with Netlify's static hosting model.

3.3 Netlify Deployment Steps

1. Open **Netlify Dashboard**.
2. Select **“Add new site” → “Deploy manually”**.
3. Drag and drop the entire STUDYPILOT project folder.
4. Netlify automatically assigns a public subdomain.
5. Optional: Site name can be customized via *Site Settings → Site Name*.
6. Deployment completes instantly, and the application becomes available via HTTPS.

Since the project contains only static files, no build script or server configuration was required.

3.4 Spotify PKCE Authorization Setup

Spotify Web API requires the **Authorization Code Flow with PKCE** for client-side applications.
The following steps were performed:

1. Generated a random **code_verifier** in the browser.
2. Produced a SHA-256 derived **code_challenge**.

Registered the Netlify URL as the official Redirect URI:

https://cool-zuccutto-dd7c3c.netlify.app/index.html

3. Implemented token exchange and authorization fully on the client, without storing sensitive information.

This ensures full security and compliance with Spotify API policies.

---

4. Configuration Plan

4.1 Final Configuration Summary

Component	Configuration
Weather API	API key + 10-minute refresh interval
Spotify API	PKCE OAuth + authorized redirect URI
LocalStorage	Max 1 MB for tasks, analytics, timer data
UI Responsiveness	320–1600 px support
Performance	Dashboard load time ≤ 2 seconds
Timer	Drift < 100 ms per Pomodoro cycle

4.2 External API Configurations

- **Spotify Web API:** Requires OAuth PKCE, redirect URI, and token refresh handling
- **OpenWeatherMap API:** Requires API key, location parameters, and error fallback

---

4.3 UI & Performance Configurations

- All UI components optimized for mobile and desktop
- JavaScript modules structured for minimal loading overhead
- Animated effects (starfield) configured to run efficiently

---

5. Environment Details

- **Browsers:** Chrome, Edge, Safari, Firefox
- **Devices:** Laptop, tablet, mobile phone
- **Network:** Internet connection required for Spotify & Weather API
- **Storage:** All user data persists in LocalStorage
- **Security Model:** No backend; no user passwords stored

---

6. Logs and Outputs

Logs generated during the deployment and runtime of the StudyPilot application capture important information such as API activity, module initialization, errors, and system warnings. These logs help verify that each component of the application is functioning as expected in the deployed environment.

StudyPilot's logs are primarily generated through browser console output (client-side JavaScript), as the application does not include a backend server. The following types of logs were collected:

- **Initialization Logs:**
  Logs produced when UI components, modules, and services are loaded (e.g., TaskManager, Pomodoro engine, Analytics module).

- **API Interaction Logs:**
  Messages confirming successful communication with external services such as Spotify Web API (PKCE authorization) and OpenWeatherMap API.

- **User Interaction Logs:**
  Events triggered by user actions, including task creation, timer start/pause, playlist selections, and weather refreshes.

- **Error and Warning Logs:**
  Any issues encountered during API calls, authorization attempts, or invalid inputs detected by the application.

Outputs are generated from functional tests, user interactions, and API responses recorded during the deployment validation process. These outputs help confirm that the deployed version of StudyPilot behaves consistently with the acceptance criteria.

Key outputs include:

- **Task Management Outputs:**
  Confirmation that tasks were successfully added, updated, deleted, and persisted in LocalStorage.

- **Pomodoro Session Outputs:**
  Results of timer accuracy testing, including drift measurements and session completion updates reflected in the analytics module.

- **Spotify Authorization Outputs:**
  Responses from the PKCE login flow showing successful token generation, playlist retrieval, and rate-limit handling.

- **Weather Recommendation Outputs:**
  Weather data retrieved from the OpenWeatherMap API and corresponding personalized study suggestions shown on the dashboard.

All logs and outputs were stored through the browser's developer tools and used during the acceptance testing phase to ensure correct system behavior.