To build a News Aggregation and Monetization website using the specified technologies (HTML/CSS/JavaScript for frontend, Node.js for backend, MySQL for the database, and News API), here's a detailed step-by-step guide:

Step 1: Setting Up the Project

1. **Install Required Tools**:
   - Install Node.js.
   - Install a code editor like Visual Studio Code.
   - Install MySQL and set up a local instance (e.g., XAMPP).

2. **Initialize the Project**:
   - mkdir NewsAggregationWebsite
   - cd NewsAggregationWebsite
   - npm init –y

   This creates a package.json file.

3. **Install Required Node.js Packages**:

   ```
   npm install express mysql2 dotenv axios cors
   ```

4. **Set Up Project Structure**:

```
NewsAggregationWebsite/
├── backend/
│   ├── app.js
│   ├── routes/
│   │   └── news.js
│   ├── controllers/
│   │   └── newsController.js
│   └── config/
│       └── db.js
├── frontend/
│   ├── index.html
│   ├── css/
│   │   └── styles.css
│   └── js/
│       └── script.js
├── .env
└── README.md
```

Step 2: Backend Development

1. **Database Configuration**:
   - Create a database named news_website in MySQL.
   - Create tables:

```
CREATE TABLE articles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255),
    description TEXT,
    url VARCHAR(255),
    category VARCHAR(50),
    published_at DATETIME
);
```

   - Configure the database connection in backend/config/db.js:

```
const mysql = require('mysql2');
const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'your_password',
    database: 'news_website'
});

connection.connect((err) => {
    if (err) throw err;
    console.log("Database connected!");
});

module.exports = connection;
```

2. **News API Integration**:
   - Fetch articles from News API using Axios.
   - backend/controllers/newsController.js:

```
const axios = require('axios');
const db = require('../config/db');

const fetchNews = async (req, res) => {
    try {
        const response = await axios.get('https://newsapi.org/v2/top-headlines', {
            params: {
                apiKey: process.env.NEWS_API_KEY,
                category: req.query.category || 'general',
                country: 'us'
            }
        });

        const articles = response.data.articles;
```

```
        articles.forEach(article => {
            const { title, description, url, publishedAt } = article;
            db.query(
                'INSERT INTO articles (title, description, url, category, published_at)
VALUES (?, ?, ?, ?, ?)',
                [title, description, url, req.query.category || 'general', publishedAt],
                (err) => {
                    if (err) console.error(err);
                }
            );
        });

        res.json(articles);
    } catch (error) {
        console.error(error);
        res.status(500).send("Error fetching news");
    }
};

module.exports = { fetchNews };
```

3. **Express Routes**:
   - Set up routes in backend/routes/news.js:

     ```
     const express = require('express');
     const router = express.Router();
     const { fetchNews } = require('../controllers/newsController');

     router.get('/fetch', fetchNews);

     module.exports = router;
     ```

   - Add route to app.js:

     ```
     const express = require('express');
     const app = express();
     const newsRoutes = require('./routes/news');
     require('dotenv').config();

     app.use('/api/news', newsRoutes);

     app.listen(3000, () => console.log('Server running on http://localhost:3000'));
     ```

Step 3: Frontend Development

1. **HTML Structure**:
   - frontend/index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>News Aggregation Website</title>
    <link rel="stylesheet" href="css/styles.css">
</head>
<body>
    <header>
        <h1>News Aggregator</h1>
        <nav>
            <input type="text" id="search" placeholder="Search news">
        </nav>
    </header>
    <main id="news-container"></main>
    <footer>
        <p>© 2024 News Aggregation Website</p>
    </footer>
    <script src="js/script.js"></script>
</body>
</html>
```

2. **CSS Styling**:
   - frontend/css/styles.css:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

header {
    background-color: #333;
    color: #fff;
    padding: 1rem;
    text-align: center;
}

#news-container {
    padding: 1rem;
    display: flex;
    flex-wrap: wrap;
    gap: 1rem;
}
```

3. **JavaScript for Frontend**:
   - `frontend/js/script.js`:

```
document.addEventListener('DOMContentLoaded', async () => {
    const response = await fetch('http://localhost:3000/api/news/fetch');
    const articles = await response.json();

    const newsContainer = document.getElementById('news-container');
    articles.forEach(article => {
        const articleDiv = document.createElement('div');
        articleDiv.className = 'article';
        articleDiv.innerHTML = `
            <h2>${article.title}</h2>
            <p>${article.description}</p>
            <a href="${article.url}" target="_blank">Read More</a>
        `;
        newsContainer.appendChild(articleDiv);
    });
});
```

Step 4: Testing and Deployment

1. **Test Locally**:
   - Start the backend: `node backend/app.js`.
   - Open `frontend/index.html` in a browser.
2. **Deploy the Backend**:
   - Use Heroku or Vercel to deploy your backend.
3. **Deploy Frontend**:
   - Use Netlify or GitHub Pages.
4. **Secure the Site**:
   - Add SSL certificates for secure connections.

Step 5: Integrate Ads and Analytics

- Add Google AdSense code to `index.html`.
- Use Google Analytics for monitoring traffic.

This plan aligns with the requirements outlined in the PDF. Let me know if you need further customization!