To become a full-stack web developer, especially if you have a background in mechanical engineering, you will need to cover a range of technologies and skills across both front-end and back-end development. Here's a structured plan, along with an estimated time frame for each step. Note that the actual time can vary based on your prior experience, learning pace, and dedication.

1. Basic Web Development Foundations (1-2 months)

- **HTML**: Learn the structure of web pages.
- **CSS**: Learn how to style web pages.
- **JavaScript**: Learn the basics of programming and interactivity on web pages.

2. Version Control (1-2 weeks)

- **Git**: Learn version control basics, how to commit changes, branch, merge, and use GitHub.

3. Front-End Development (2-3 months)

- **Advanced JavaScript**: Dive deeper into ES6+ features, asynchronous programming, and DOM manipulation.
- **Front-End Frameworks/Libraries**: Learn a popular front-end library or framework like React, Angular, or Vue.js.
- **Responsive Design**: Learn about responsive design principles and frameworks like Bootstrap or Tailwind CSS.

4. Back-End Development (3-4 months)

- **Server-Side Programming**: Learn a back-end language such as Node.js (JavaScript), Python (Django or Flask), Ruby (Rails), or Java (Spring).
- **Databases**: Understand relational databases (SQL) like PostgreSQL or MySQL, and NoSQL databases like MongoDB.
- **APIs**: Learn how to create and consume RESTful and GraphQL APIs.

5. Full-Stack Development (2-3 months)

- **Integration**: Learn how to connect front-end and back-end, handling sessions, authentication, and authorization.
- **Build a Full-Stack Project**: Create a project that includes user authentication, CRUD operations, and connects a front-end with a back-end.

6. DevOps and Deployment (1-2 months)

- **Environment Management**: Learn about environments and package managers like npm, yarn, pip.
- **Deployment**: Learn how to deploy applications using services like Heroku, AWS, Netlify, or Vercel.
- **CI/CD**: Learn about Continuous Integration/Continuous Deployment practices.

7. Soft Skills and Additional Learning (Ongoing)

- **Problem-Solving**: Enhance problem-solving skills through coding challenges and algorithms.
- **Collaboration**: Contribute to open-source projects or work on team projects to improve collaboration skills.
- **Continued Learning**: Web development is constantly evolving. Stay updated with new technologies and best practices.

Total Estimated Time: 9-14 months

Detailed Learning Path and Resources:

1. **HTML & CSS**
   - **Resources**: MDN Web Docs, freeCodeCamp, Codecademy
   - **Projects**: Build a personal portfolio website, responsive web pages
2. **JavaScript Basics**
   - **Resources**: Eloquent JavaScript, You Don't Know JS, freeCodeCamp
   - **Projects**: Interactive elements on web pages (forms, buttons)
3. **Git & GitHub**
   - **Resources**: Git documentation, GitHub Learning Lab
   - **Projects**: Version control for personal projects
4. **Advanced JavaScript & Frameworks**
   - **Resources**: React Documentation, Angular Documentation, Vue.js Guide
   - **Projects**: SPA (Single Page Applications), dynamic user interfaces
5. **Server-Side Programming**
   - **Resources**: Node.js Documentation, Django Documentation, Flask Documentation
   - **Projects**: RESTful APIs, CRUD applications
6. **Databases**
   - **Resources**: PostgreSQL, MongoDB Documentation, SQL Tutorials
   - **Projects**: Database-driven applications, data storage for projects
7. **Full-Stack Integration**
   - **Resources**: Full-stack project tutorials, MERN/MEAN stack guides
   - **Projects**: Full-stack applications with front-end and back-end integration
8. **DevOps & Deployment**
   - **Resources**: Heroku, AWS documentation, Docker tutorials
   - **Projects**: Deploying personal projects, setting up CI/CD pipelines

Tips for Success:

- **Consistent Practice**: Dedicate regular hours each week to learning and practicing.
- **Hands-on Projects**: Build real-world projects to solidify your knowledge.
- **Networking**: Join web development communities, attend meetups, and engage with peers.
- **Continuous Learning**: Keep up with industry trends and new technologies.

By following this structured approach, you can transition from a mechanical engineering background to a proficient full-stack web developer within a year.