Full-Stack Web Development

EN MUHAMMAD FAREEZ BIN BORHANUDIN
https://github.com/fare4z

NOTES & MATERIALS
https://fare4z.com/url/Ry8d7

Agenda

1. Introduction to Full-Stack Web Development
2. Version Control System – VCS
3. Front-End Development
4. Back-End Development

Full-stack Web Development

- Full stack development is the practice of being proficient in both the **front-end** and **back-end** aspects of web application development. A full stack developer is capable of working on all layers of a software application, from the user interface and user experience (front-end) to the server, database, and server-side logic (back-end). This versatility allows them to create and maintain complete web applications independently or as part of a development team.

Full-Stack Developer

Front-End	Back-End	Full-Stack
• What user see • UI & UX • HTML , JS, CSS	• Application/system logic • Database connection/manipulation • PHP, Java, Python, Node.js [Javascript] • Security	• Front-End + Back-End

Layer of Full-Stack Development

Front-End

Create the visual part of the application that users interact with.

- **Languages and Technologies:**
  - **HTML**
  - **CSS**
  - **JavaScript**

- **Frameworks/Libraries:**
  - **React.js, Angular, Vue.js:** Enhance JavaScript capabilities for building complex and efficient user interfaces.
  - **Bootstrap, Tailwind CSS:** Help with responsive design and pre-built components for faster UI development.

Back-End

Handle the application's logic, database interactions, user authentication, and server configuration.

- **Languages and Technologies:**
  - **Languages:** PHP, Python, Ruby, Java, Node.js (JavaScript).
  - **Frameworks:** CodeIgniter (PHP), Django (Python), Ruby on Rails (Ruby), Spring (Java), Express.js (Node.js).
  - **Databases:**
    - **SQL Databases:** MySQL, PostgreSQL, SQLite.
    - **NoSQL Databases:** MongoDB, CouchDB.
  - **APIs (Application Programming Interfaces)**
- **Server and Hosting:**
  - **Web Servers:** Apache, Nginx.
  - **Cloud Platforms:** AWS, Microsoft Azure, Google Cloud Platform.
  - **Containers:** Docker for creating and managing containers that hold applications and their dependencies.

Database Management

Store, manage, and retrieve application data efficiently and securely.

- **Skills:**
  - Database design, normalization, and optimization.
  - Writing complex SQL queries, stored procedures, and triggers.
  - Understanding database transactions, indexing, and performance tuning.

Version Control & DevOps

Manage and automate the deployment, monitoring, and scaling of web applications.

Tools and Practices:

- **Version Control:** Git, GitHub, GitLab for tracking changes in code and collaborating with other developers.
- **CI/CD (Continuous Integration/Continuous Deployment):** Jenkins, GitLab CI/CD, CircleCI for automating testing and deployment.
- **Containerization:** Docker for packaging applications with all their dependencies.
- **Orchestration:** Kubernetes for managing containerized applications across multiple environments.

Security

Purpose: Ensure the application is secure from threats and vulnerabilities.

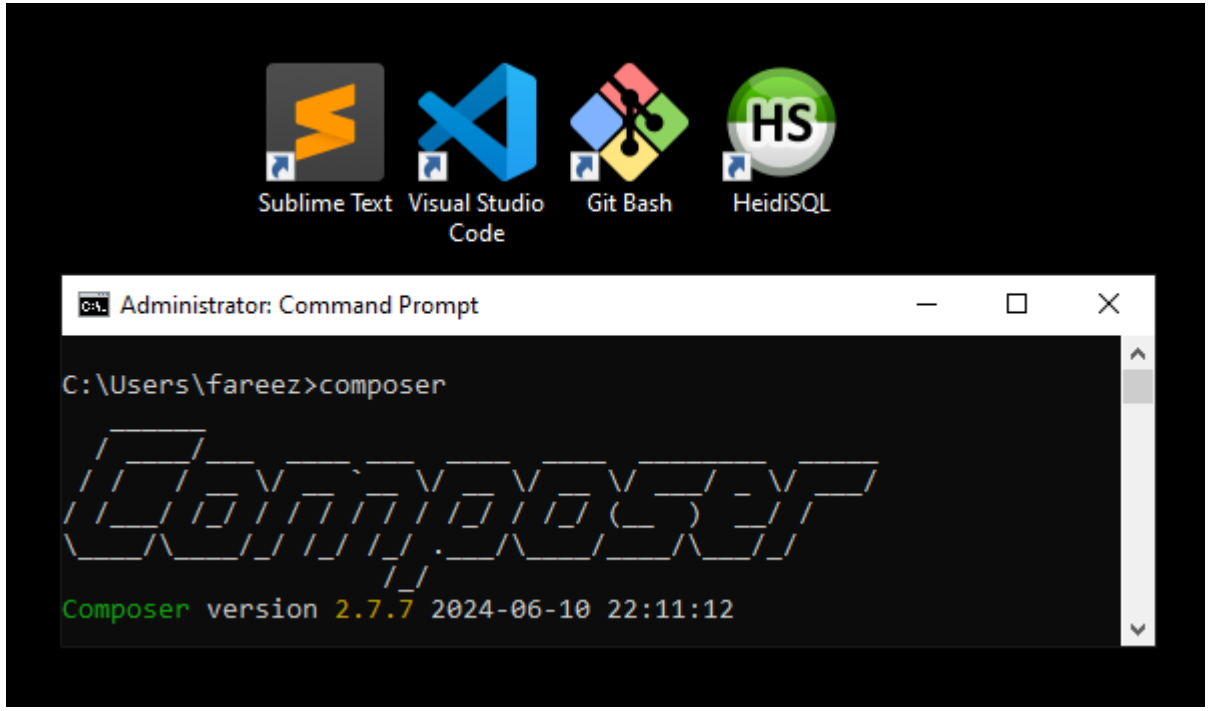
- **Practices:**
  - Implementing SSL/TLS for secure data transmission.
  - Using OAuth and JWT (JSON Web Tokens) for secure authentication and authorization.
  - Protecting against common vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

Skills Required for Full-Stack Development

- **Technical Skills:**
  - Proficiency in front-end and back-end programming languages.
  - Understanding of database management and writing complex queries.
  - Familiarity with version control systems like Git.
  - Experience with cloud platforms and server management.
  - Knowledge of RESTful services and APIs.
- **Soft Skills:**
  - **Problem-solving:** Ability to diagnose and resolve issues across different layers of the stack.
  - **Communication:** Effectively collaborating with other developers, designers, and stakeholders.
  - **Time Management:** Handling multiple tasks and projects efficiently.
  - **Adaptability:** Keeping up with rapidly evolving technologies and best practices in web development.

Required Tools

- Code Editors and IDE :-
  - VS Code , Sublime Text
  - Intellij IDEA
- Composer
  - https://getcomposer.org/download/
- Git Scm
  - https://git-scm.com/
- Database Tools
  - phpMyAdmin
  - HeidiSQL
    - https://www.heidisql.com/



Practical Activity

- Install and configure tools
- Adding the PHP folder to the windows path environment variable
  - setx path "%path%; C:\path\to\php“
  - php –v
  - git --version
  - composer

VCS

Version Control System

VCS

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
- It allows to maintain code efficiently.

Types of VCS

Local VSC

- All changes done in the file will be stored locally and will be only available on user's machine.
- Since all the changes are stored locally, If the hard disk gets corrupted then the project can be lost.

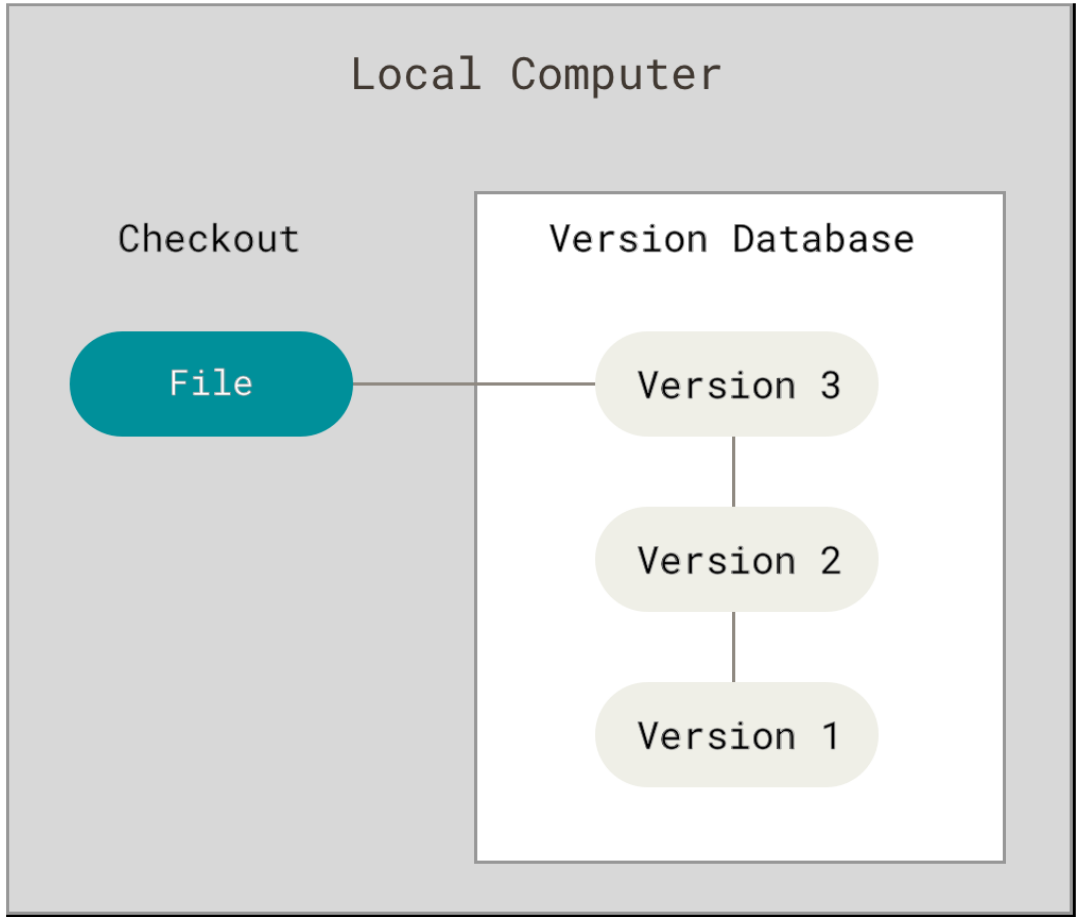
Centralized VCS

- There is only one central repository where all changes are made by multiple people. Multiple people can get the code from central repo and then push their changes into same repo.
- In case central repo is not available or down for some time no one will be able to get the code from the repo and push the changes.

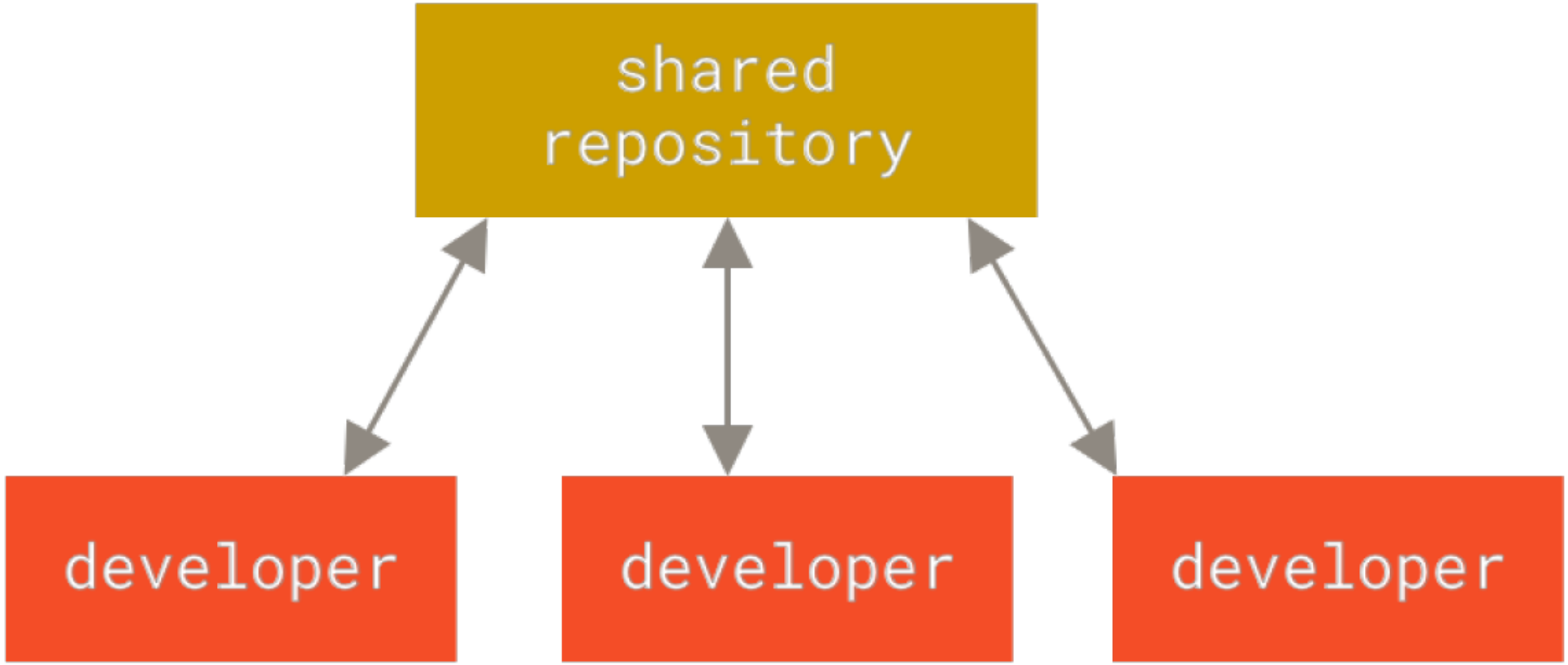
Distributed VCS

- User has their own copy of repository from central repository.
- Even if central repo goes down user still has their own copy of repo and they can continue working with that.

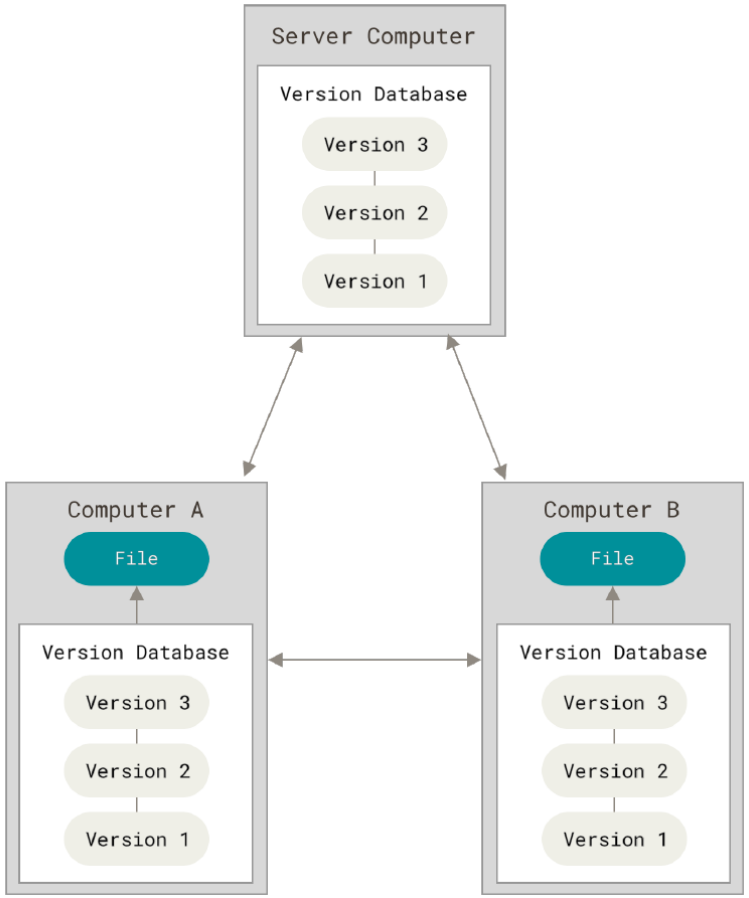
Local VCS



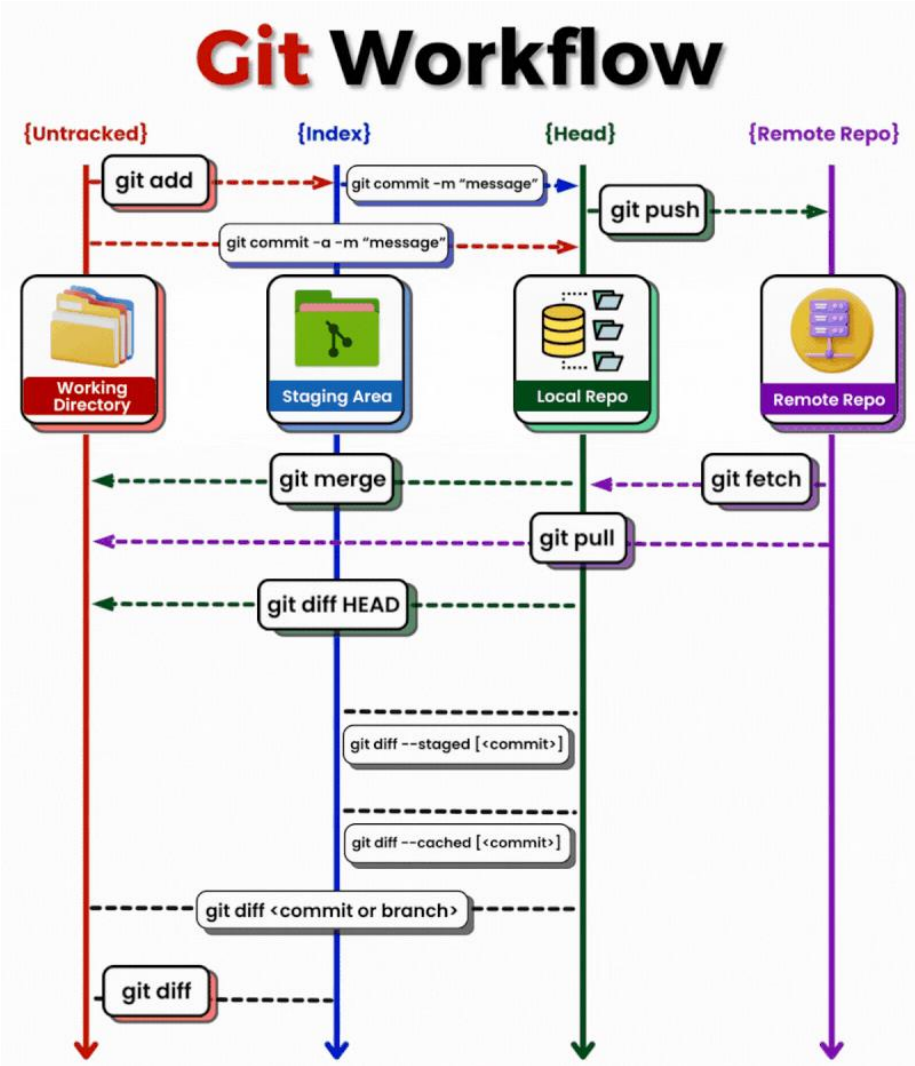
Centralized VCS



Distributed VCS



Git



- Git is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers who are collaboratively developing source code during software development.

Common Term in Git

- **Repository (Repo)**
  - A repository is a storage space where your project files and their history are stored. It can be local (on your computer) or remote (on GitHub).
- **Commit**
  - A commit is a snapshot of your project's files at a specific point in time. It saves changes to the repository with a message describing what was changed.
- **Branches**
  - Branches are used to develop features, fix bugs, or experiment in a contained area of your project. The default branch in Git is called main or master.
- **Pull**
  - Fetch and download content from a remote repository and immediately update the local repository to match that content [Download]
- **Push**
  - Transfer commits from your local repository to a remote repo [Upload]
- **Clone**
  - Make a copy of Repo

Basic Git Command

- **Menyemak Versi Git**
  - git --version
- **Menetapkan Pengguna**
  - git config --global user.name "NAMA"
  - git config --global user.email "alamat@email"
- **Menyemak Status Git**
  - git status
- **Bertukar Ke Branch atau Commit lain**
  - git checkout {nama branch}
  - git checkout {ID commit}
- **Menambah Remote Repo**
  - git remote add origin {URL remote repo}
- **Membuat Branch Baru Dari Branch Semasa**
  - git checkout -b {nama branch}
- **Menambah File Untuk Staging (Sebelum Commit)**
  - git add {nama fail}
  - git add .
- **Push Ke Remote Repo**
  - git push origin {nama branch}
- **Pull Dari Remote Repo**
  - git pull origin {nama branch}

Practical Activity

- Create a GitHub account: https://github.com/
- Apply for GitHub Education: https://github.com/education
- Create a repository locally and on GitHub
- Make your first commit
- Push changes to GitHub
- Publish to GitHub Pages

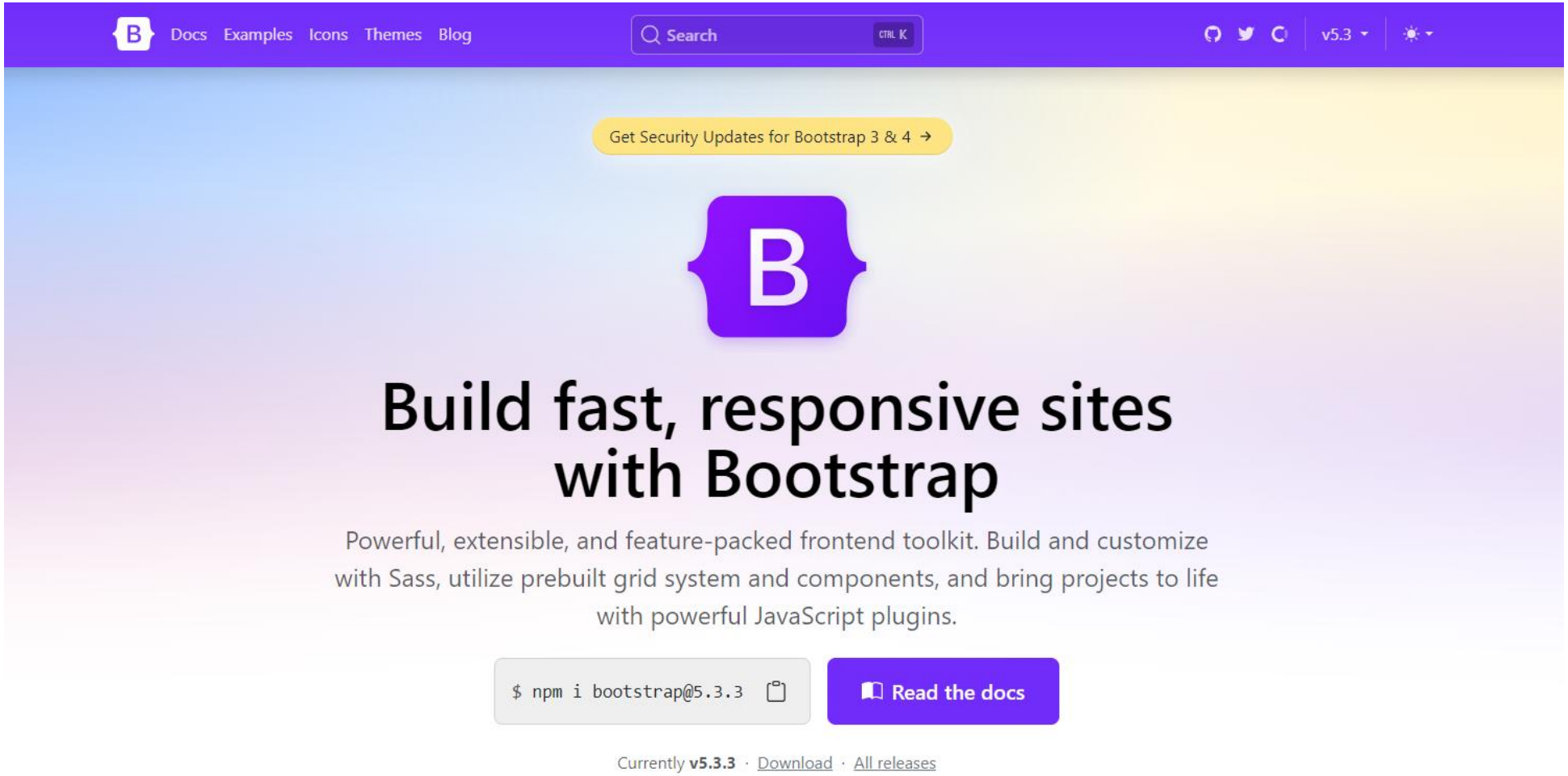
Front End Development

Bootstrap CSS Framework

Recap Web Design Technology …

- **HTML (HyperText Markup Language)**
  - **Purpose**: Structure of web pages.
  - **Key Elements**: Headings, paragraphs, links, images.
- **CSS (Cascading Style Sheets)**
  - **Purpose**: Styling web pages.
  - **Example**: Colors, Fonts, Layouts.
- **JavaScript**
  - **Purpose**: Adding interactivity.
  - **Key Concepts**: Variables, Functions, Events.
  - **Libraries/Frameworks**: jQuery, React, Vue.

CSS Framework



Basic Text Utilities

1) Text Alignment
2) Text Transformation
3) Text Weight and Italic
4) Text Color
5) Text Break and Wrap
6) Muted and Lead Text

Text Alignment

```
<p class="text-start">This is left-aligned text.</p>
```

Class	Description
.text-start	Left-aligned text
.text-center	Center-aligned text
.text-end	Right-aligned text

Text Transformation

```
<p class="text-uppercase">This text will appear in uppercase.</p>
```

Class	Description
.text-uppercase	Changes to uppercase
.text-lowercase	Changes to lowercase
.text-capitalize	Changes the first letter of each word

Text Weight and Italic

```
<p class="fw-bold">This is bold text.</p>
```

Class	Description
.fw-bold	Bold text
.fw-bolder	Bolder weight text (relative to the parent element)
.fw-semibold	Semibold weight text
.fw-medium	Medium weight text
.fw-normal	Normal weight text
.fw-light	Light weight text
.fw-lighter	Lighter weight text (relative to the parent element)
.fst-italic	Italic text
.fst-normal	Text with normal font style

Text Color/Utilities

```
<p class="text-primary">This text is blue (primary).</p>
```

Class	Description
.text-primary	Blue (default primary color)
.text-secondary	Gray
.text-success	Green
.text-danger	Red
.text-warning	Yellow
.text-info	Cyan
.text-light	Light gray (best used on dark backgrounds)

Class	Description
.text-dark	Dark gray (best used on light backgrounds)
.text-body	Default body text color
.text-muted	Muted (light gray) text
.text-white	White text (best used on dark backgrounds)
.text-black-50	50% opaque black text
.text-white-50	50% opaque white text
.lead	Set the text to a larger font size than the default

Text Break and Wrap

These classes are often used in conjunction with other CSS properties like width and max-width to control the layout and appearance of text within a container.

Class	Description
.text-wrap	allows text to wrap onto the next line when it reaches the end of a container.
.text-break	forces text to break at word boundaries

Layout and Spacing

1) Container – Basic Layout Wrapper
2) Spacing – Margin and Padding
3) Grid System – Dividing content into column

Container

```
<div class="container">Content inside a responsive fixed-width container.</div>
```

Class	Description
.container	A responsive fixed-width container that adapts to the screen size.
.container-fluid	A full-width container that takes up 100% of the viewport width.
.container-{breakpoint}	A container that’s responsive and only takes up fixed width on specified breakpoints (e.g., .container-md)

Spacing

{property}{sides}-{breakpoint}-{size}

Property	Sides	Description	Size
m	t	Sets margin-top	0
p	b	Sets padding-bottom	1
	s	Sets margin-left (LTR) or padding-left (RTL)	2
	e	Sets margin-right (LTR) or padding-right (RTL)	3
	x	Sets both left and right margins/padding	4
	y	Sets both top and bottom margins/padding	5
	(blank)	Applies the margin or padding to all four sides	auto

Breakpoint

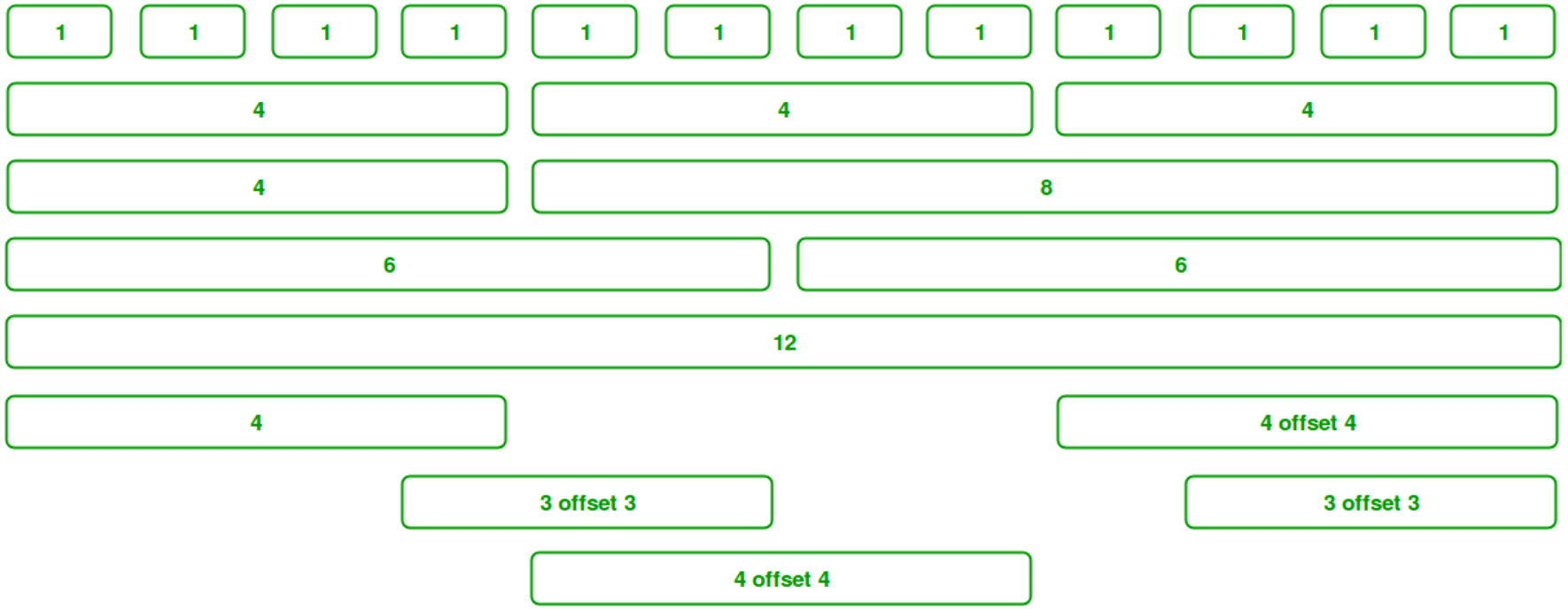
Breakpoints are customizable widths that determine how your responsive layout behaves across device or viewport sizes in Bootstrap.

Breakpoint	Class infix	Dimensions
Extra small	None	<576px
Small	sm	≥576px
Medium	md	≥768px
Large	lg	≥992px
Extra large	xl	≥1200px
Extra extra large	xxl	≥1400px

Grid

- The grid system in Bootstrap is based on 12 columns.
- It allows content to be arranged into rows and columns.
- You can control the width of the columns by defining how many out of 12 columns they span.
- Breakpoints help make the grid responsive for different screen sizes

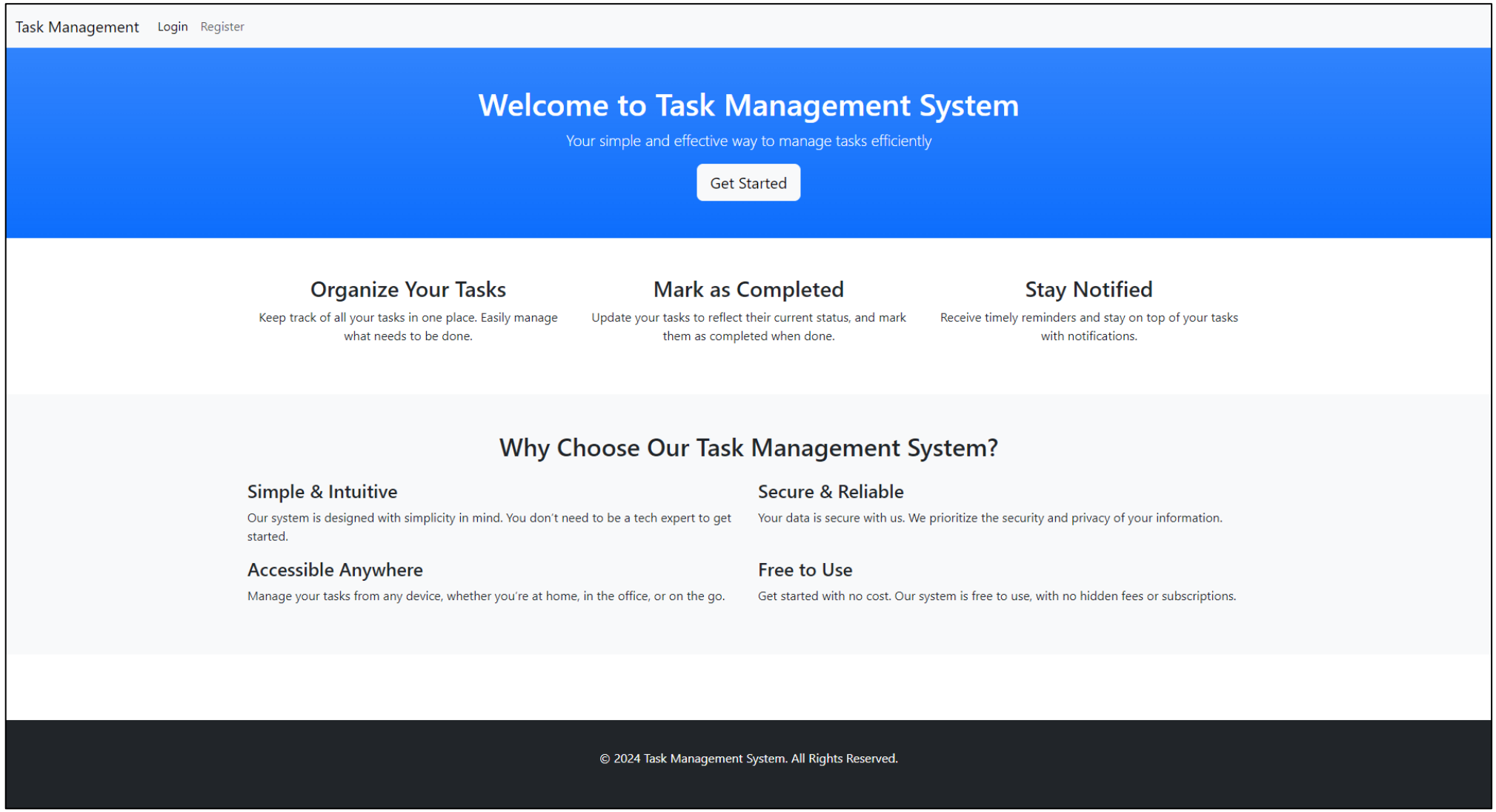
```
<div class="col-6">50% width on all screen sizes.</div>
<div class="col-md-6">6 columns on medium screens and larger.</div>
```



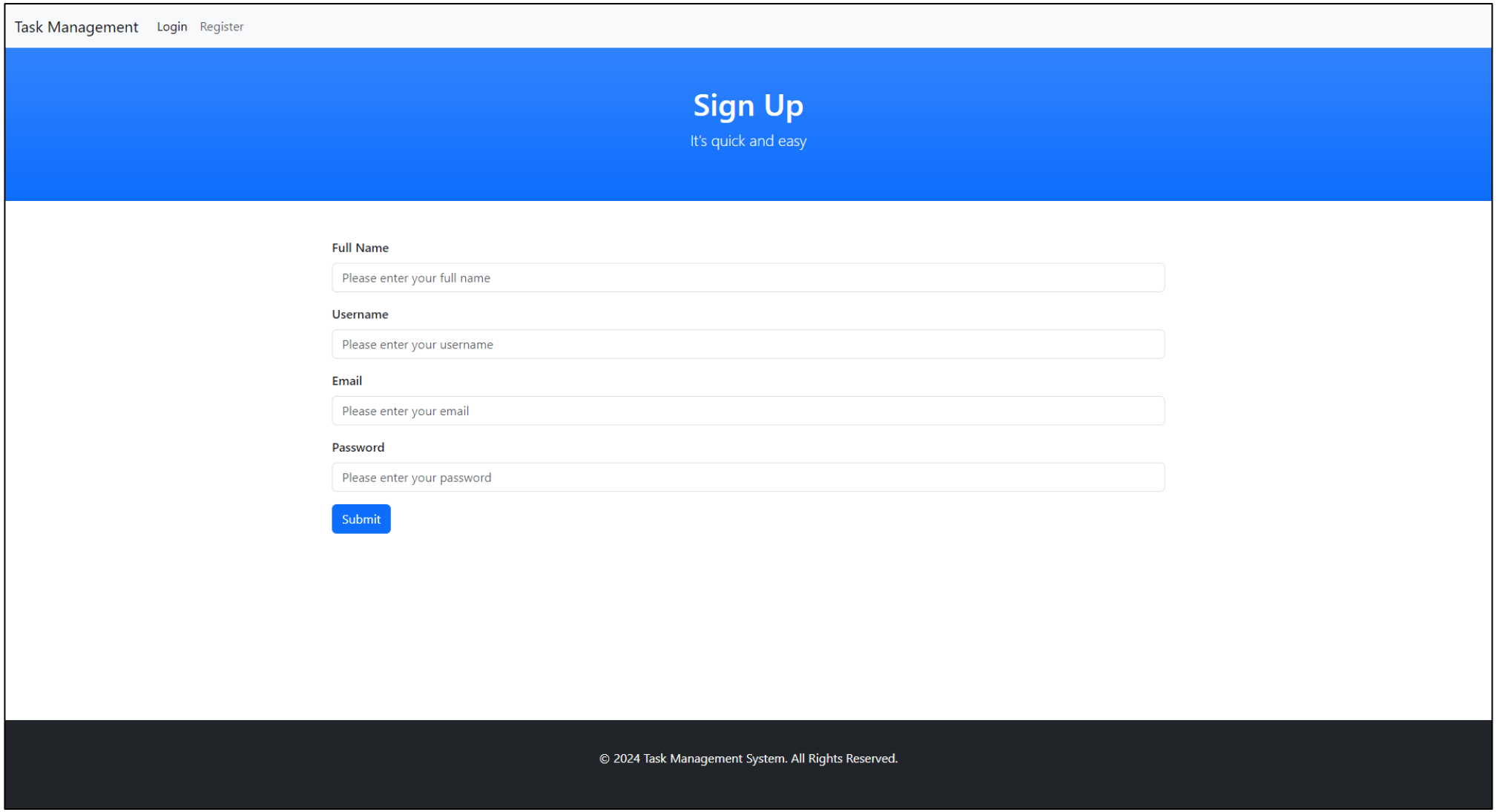
Lab Activity

You are required to develop a Task Management System user interface using Bootstrap. Refer to the provided screenshots for layout and design inspiration. Once completed, push your project to your GitHub account.

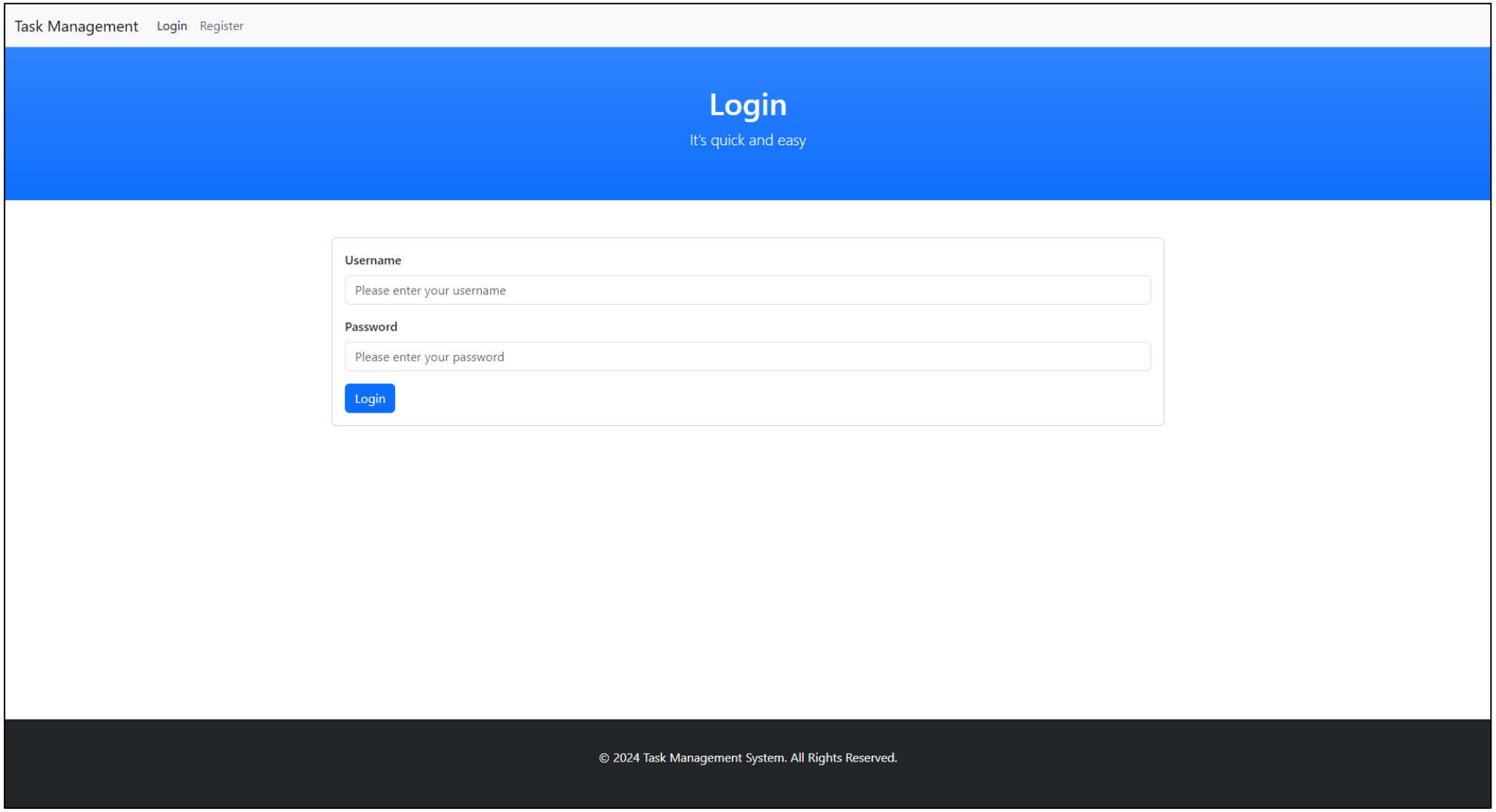
main.html



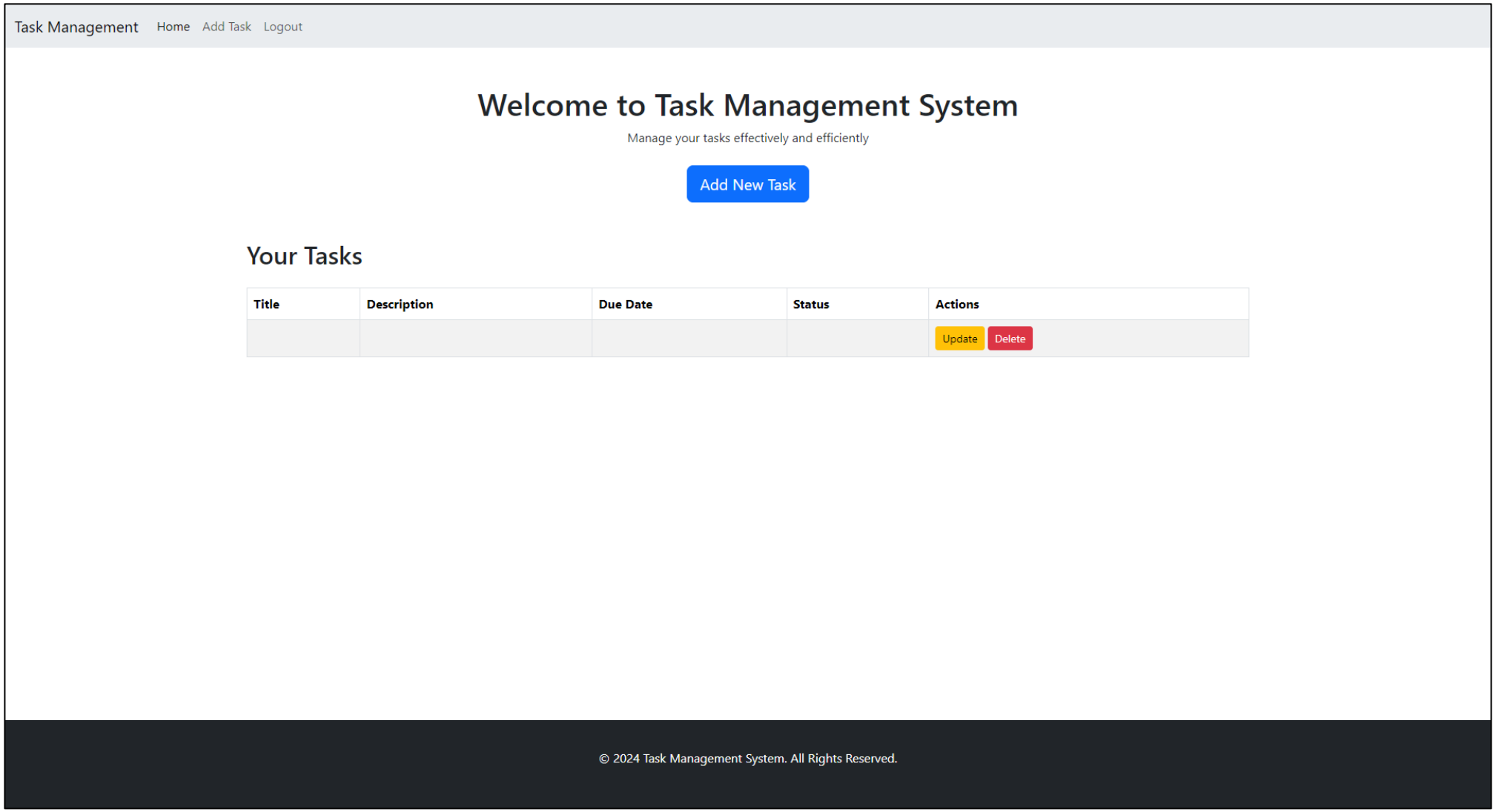
register.html



login.html



dashboard.html



newtask.html