**2025-3-6 COMPSC732 Summary**

**英文原文 & 中文翻译**

1. 课程概览（Course Overview）

（中文）

• 这门课在研究生阶段的计算机科学课程中规模最大，目前约有 309 人注册。

• 主要内容涵盖前端框架（React）以及相关后端技术（通常称为 MERN 栈：MongoDB、Express、React、Node.js），并注重项目实践。

• 课程包括三大部分：作业（Assignment）、测试（Quiz）、以及项目（Project）。

• 教师对同学的期望包括：

1. 安装所需环境（Node.js、VS Code、Git 等），并熟悉基本使用方法。

2. 充分利用提供的在线资源（Canvas 模块、小视频、Ed Discussion 等）。

3. 参与实验课（Labs）以获取对项目或课程作业的帮助。

4. 平均每周投入约 10 小时，但根据实际进度会有所波动。

（英文）

• This is the largest postgrad Computer Science course with about 309 students currently enrolled.

• The main content covers frontend frameworks (React) and related backend technologies (often referred to as the MERN stack: MongoDB, Express, React, Node.js), with a strong focus on project-based practice.

• The course comprises three major components: the Assignment, the Quiz, and the Project.

• Instructor expectations include:

1. Installing required tools (Node.js, VS Code, Git, etc.) and being comfortable with basic usage.

2. Making full use of the provided online resources (Canvas modules, short videos, Ed Discussion, etc.).

3. Attending lab sessions (Labs) for assistance with projects or assignments.

4. Spending around 10 hours per week on average, though this may vary as deadlines approach.

2. 作业和评估（Assignments and Assessments）

2.1 作业（Assignment）

（中文）

• 个人作业，占比 30%。

• 需研究一个本课程未覆盖的 Web 技术，并制作一个展示视频（Tech Demo）。

• 如果你之前在其他课程中已使用过某技术，则不可再次用同样技术完成本作业。

• 例如，如果你在其他课里做过 Svelte，则本次作业不能再选 Svelte。

• 互评环节：你需要观看并评价其他同学的视频，这部分互评成绩占你课程总分的 2%。

• 建议选择对你项目或职业规划有帮助的技术方向进行研究。可在作业说明中寻找灵感或自行与老师沟通想法。

（英文）

• This is an individual assignment, worth 30% of your grade.

• You must research a web technology not covered in-depth by this course, and produce a demo video (Tech Demo).

• You cannot reuse a technology already covered in a different course you took (e.g., no Svelte again if you’ve done it before).

• Peer review: you will watch and evaluate other students’ demo videos. This peer-review component is worth 2% of your final mark.

• It is advisable to choose a technology that could benefit your project or future career path. You can refer to the assignment guidelines or consult with the instructor about your own ideas.

2.2 测试（Quiz）

（中文）

• 形式类似“take-home”测验，会在约 24 小时的时间窗内开放，一旦开始则需在 3 小时内完成。

• 测试重点在于 MERN 栈的个人实际掌握程度。

• 可以使用 AI 工具（如 Copilot）和其他公开资源，但需在测验提交时说明使用情况。

• 难度不会太高，但要合理规划时间并熟悉基本的全栈开发流程。

（英文）

• The quiz is a “take-home” style assessment, open for about 24 hours, but once started, you have 3 hours to complete it.

• It mainly tests your individual proficiency with the MERN stack.

• AI tools (e.g., Copilot) and public resources are permitted, but you must disclose how you used them when you submit.

• The difficulty is manageable, yet time management and familiarity with full-stack fundamentals are crucial.

2.3 项目（Project）

（中文）

• 小组项目（6 或 7 人/组），占比最高，含多个阶段性产出：

1. 项目提案（Proposal）：简要说明项目想法、可行性、风险评估等（全组同分）。

2. 项目实现（Implementation）：实际的代码与功能实现（以团队为单位打分，但会考虑个人贡献度）。

3. 项目演示（Presentation）：在学期后期进行研讨会形式的展示；时间段先到先得。

4. 互评（Peer Review）：你需评审其他团队的项目成果。

5. 个人报告（Report）：每位同学独立撰写，对自己在项目中的贡献、反思等进行总结。

• 选题可参考 Canvas 提供的示例，也可自由发挥，但需掌握好范围和复杂度。

• 鼓励使用团队协作工具（GitHub、Trello、Jira 等），并采用敏捷或其他项目管理模式（Scrum、Kanban 等）。

（英文）

• A team-based project (6 or 7 members per team), carrying the largest weight, with multiple deliverables:

1. Project Proposal: a brief outline of your idea, feasibility, and risk assessment (teamwide score).

2. Implementation: actual code and functionality (team grade, but individual contributions are considered).

3. Presentation: a seminar-style demo in the later weeks of the semester; slots are first-come, first-served.

4. Peer Review: you will evaluate another team’s project deliverables.

5. Individual Report: each member writes an individual reflection and summary of their contributions.

• You can reference example topics on Canvas, or propose your own, but be mindful of scope and complexity.

• Team collaboration tools (GitHub, Trello, Jira, etc.) and an agile or other project management process (Scrum, Kanban, etc.) are highly recommended.

3. React 重点与扩展（Key Points in React, with Extensions）

3.1 React 基础与生态

（中文）

• React 最早于 2012/2013 年推出，由 Facebook（现 Meta）维护，目前仍是使用最广泛的前端框架之一。

• 强调组件化与状态管理，通过声明式编程提高开发效率。

• React 生态非常庞大：除了 React 核心之外，还包括 React Router（路由管理）、Redux/MobX/Recoil（进阶状态管理）以及 Next.js（SSR 和路由自动化）等。

• 本课程主要关注“React + Node.js + Express + MongoDB”构成的 MERN 栈，但学完后扩展到其他框架也相对容易。

（英文）

• Originally introduced around 2012/2013, React is maintained by Facebook (now Meta) and remains one of the most widely used frontend frameworks.

• It emphasizes component-based architecture and state management, leveraging declarative programming to boost productivity.

• React has a vast ecosystem beyond just the core library, including React Router for routing, Redux/MobX/Recoil for advanced state management, and Next.js for SSR and automated routing, among others.

• This course will mainly focus on the MERN stack (“React + Node.js + Express + MongoDB”), but once you master React’s fundamentals, exploring other frameworks or libraries becomes much easier.

3.2 React Hooks：useState 与 useEffect

（中文）

1. useState

• 用于在函数组件中管理本地状态，并触发重新渲染。

• 常见用法：

import { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0);

return (

<div>

<p>当前计数：{count}</p>

<button onClick={() => setCount(count + 1)}>增加</button>

</div>

);

}

• 调用 setCount 会让 React 重新渲染组件。

2. useEffect

• 用于在组件渲染后执行副作用逻辑，如数据获取、订阅或清理等。

• 若依赖数组留空 []，则该副作用只在首次挂载后运行一次；若未设依赖数组，则每次渲染后都会运行。

• 示例：

import { useEffect, useState } from 'react';

function GreetingFetcher() {

const [greeting, setGreeting] = useState(null);

useEffect(() => {

// 仅在组件初次挂载后执行一次

fetch('/api/greeting')

.then(res => res.json())

.then(data => setGreeting(data.message))

.catch(err => console.error(err));

}, []);

return (

<div>

{greeting ? <h1>{greeting}</h1> : <p>加载中...</p>}

</div>

);

}

（英文）

1. useState

• Manages local state within function components and triggers re-rendering.

• Common usage:

import { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0);

return (

<div>

<p>Current Count: {count}</p>

<button onClick={() => setCount(count + 1)}>Increment</button>

</div>

);

}

• Calling setCount prompts React to re-render the component.

2. useEffect

• Executes side effects after the component renders, such as fetching data, subscribing to events, or performing cleanups.

• An empty dependency array [] means the effect runs only once after the initial mount; omitting the array means it runs after every render.

• Example:

import { useEffect, useState } from 'react';

function GreetingFetcher() {

const [greeting, setGreeting] = useState(null);

useEffect(() => {

// Runs only after the component mounts the first time

fetch('/api/greeting')

.then(res => res.json())

.then(data => setGreeting(data.message))

.catch(err => console.error(err));

}, []);

return (

<div>

{greeting ? <h1>{greeting}</h1> : <p>Loading...</p>}

</div>

);

}

3.3 CSS Modules 与组件化样式

（中文）

• 在 React 中，我们可使用 CSS Modules 实现样式的局部作用域，避免全局命名冲突。

• 文件命名约定通常为 MyComponent.module.css，使用时先 import styles from './MyComponent.module.css';，再用 className={styles.someClass} 绑定样式。

• 示例：

/\* Greeting.module.css \*/

.flag {

width: 200px;

}

import React from 'react';

import styles from './Greeting.module.css';

export default function Greeting({ greeting }) {

return (

<div>

<img className={styles.flag} src={greeting.flagUrl} alt="flag" />

<h2>{greeting.text}</h2>

</div>

);

}

（英文）

• In React, you can use CSS Modules for locally scoped styles, preventing global naming conflicts.

• The file naming convention is usually MyComponent.module.css. You then import styles from './MyComponent.module.css' and bind styles using className={styles.someClass}.

• Example:

/\* Greeting.module.css \*/

.flag {

width: 200px;

}

import React from 'react';

import styles from './Greeting.module.css';

export default function Greeting({ greeting }) {

return (

<div>

<img className={styles.flag} src={greeting.flagUrl} alt="flag" />

<h2>{greeting.text}</h2>

</div>

);

}

3.4 第三方 UI 库与组件设计

（中文）

• 常见的 React UI 库有 Material UI (MUI)、Ant Design、Chakra UI 等，可显著加快界面开发。

• 这些库提供丰富的可复用组件（Button、Modal、Dialog、Table 等），并带有一致的设计风格。

• 在实习或项目中，如果需要更快迭代，可以优先考虑使用这些成熟组件。若对设计有高要求，可在其基础上自定义主题。

• 例如使用 MUI：

npm install @mui/material @emotion/react @emotion/styled

import Button from '@mui/material/Button';

function MyComponent() {

return (

<Button variant="contained">点击我</Button>

);

}

（英文）

• Popular React UI libraries include Material UI (MUI), Ant Design, and Chakra UI, significantly speeding up interface development.

• They provide a variety of reusable components (Buttons, Modals, Dialogs, Tables, etc.) with a cohesive design style.

• For faster iteration in a project or internship, consider leveraging these well-maintained libraries. If you have stricter design requirements, you can customize themes accordingly.

• Example with MUI:

npm install @mui/material @emotion/react @emotion/styled

import Button from '@mui/material/Button';

function MyComponent() {

return (

<Button variant="contained">Click Me</Button>

);

}

4. 后端与数据库扩展（Backend and Database Extensions）

（中文）

• 本课通常使用 Node.js + Express 搭建后端，并使用 MongoDB 存储数据。

• 你需要熟悉基本的 RESTful API 设计，如 GET /api/items、POST /api/items 等，对应在 Express 中编写路由和中间件。

• 推荐在实际项目中使用 Mongoose 来简化对 MongoDB 的操作，并在路由层进行数据验证，避免错误数据进入数据库。

• 样板代码示例：

// server.js (简化示例)

const express = require('express');

const mongoose = require('mongoose');

const app = express();

mongoose.connect('mongodb://localhost:27017/demo', { useNewUrlParser: true })

.then(() => console.log('MongoDB connected'))

.catch(err => console.error(err));

app.use(express.json());

// 定义一个简单的 Schema

const itemSchema = new mongoose.Schema({

name: String,

price: Number

});

const Item = mongoose.model('Item', itemSchema);

// 路由示例

app.get('/api/items', async (req, res) => {

const items = await Item.find();

res.json(items);

});

app.post('/api/items', async (req, res) => {

const newItem = new Item(req.body);

await newItem.save();

res.status(201).json(newItem);

});

app.listen(3000, () => {

console.log('Server running on port 3000');

});

（英文）

• The typical stack here is Node.js + Express on the backend, with MongoDB as the database.

• You should be comfortable designing basic RESTful APIs, e.g., GET /api/items, POST /api/items, etc., by implementing routes and middleware in Express.

• Using Mongoose is recommended to simplify MongoDB operations; also consider validating data at the route level to prevent invalid entries.

• Sample code snippet:

// server.js (simplified example)

const express = require('express');

const mongoose = require('mongoose');

const app = express();

mongoose.connect('mongodb://localhost:27017/demo', { useNewUrlParser: true })

.then(() => console.log('MongoDB connected'))

.catch(err => console.error(err));

app.use(express.json());

// Define a simple schema

const itemSchema = new mongoose.Schema({

name: String,

price: Number

});

const Item = mongoose.model('Item', itemSchema);

// Example routes

app.get('/api/items', async (req, res) => {

const items = await Item.find();

res.json(items);

});

app.post('/api/items', async (req, res) => {

const newItem = new Item(req.body);

await newItem.save();

res.status(201).json(newItem);

});

app.listen(3000, () => {

console.log('Server running on port 3000');

});

5. 其他补充（Additional Notes）

（中文）

• 抄袭与 AI 工具使用：

• 可以使用 AI 辅助写代码，但要与团队沟通，确保可维护。

• 切勿让 AI 代写报告或作业，否则质量与真实性均难以保证。

• 提交测试或作业时，若使用 AI，请如实说明。

• 时间管理：

• 每周留够时间学习和实验，尽量提前开始项目，尤其是团队合作要协调时间。

• 团队沟通：

• 建议在团队中明确分工，但保持每个人对全局有一定了解。

• 遇到技术难题可在 Ed Discussion、实验课或邮件中咨询老师与助教。

（英文）

• Plagiarism and AI Tools:

• AI can assist in coding, but coordinate with your team to maintain code quality.

• Avoid relying on AI for writing reports or assignments; the content might be low-quality or lack authenticity.

• If AI tools are used for tests or assignments, disclose it honestly.

• Time Management:

• Allocate sufficient weekly time for both study and lab work, and start the project early.

• Coordination is essential in team projects.

• Team Communication:

• Clarify roles, but ensure each member has some overview of the entire system.

• For any technical challenges, reach out via Ed Discussion, labs, or email to the instructor and TAs.