

# **Happy Go Lucky Introduction**



Dirk Riehle, FAU Erlangen

**ADAP C01**

Licensed under [CC BY 4.0 International](https://creativecommons.org/licenses/by/4.0/)

# Happy Go Lucky Vision



Happy Go Lucky (HGL) is

- A web app to support our project based teaching

Original HGL solutions are

- Happiness index
- Standup emails
- Code tracking

# Setup and Test of Happy Go Lucky

Fork happy-go-lucky to your account, e.g. friedalex

```
git clone git@github.com:friedalex/happy-go-lucky.git
```

```
cd happy-go-lucky
```

```
npm install
```

```
npm run build
```

```
npm run test
```

```
npm run generate-mockdata
```

```
npm run test
```

```
npm run dev
```

# Happy Go Lucky Base Design



HGL is a web app to support our project based teaching

An admin (professor) can create courses (by semester)

Courses have an associated schedule (homework delivery dates)

Users (anyone) can create projects and add them to courses

A project can have one or more members (ADAP = 1, AMOS = 6..12)

A project is linked to exactly one GitHub repository

# ADAP and AMOS



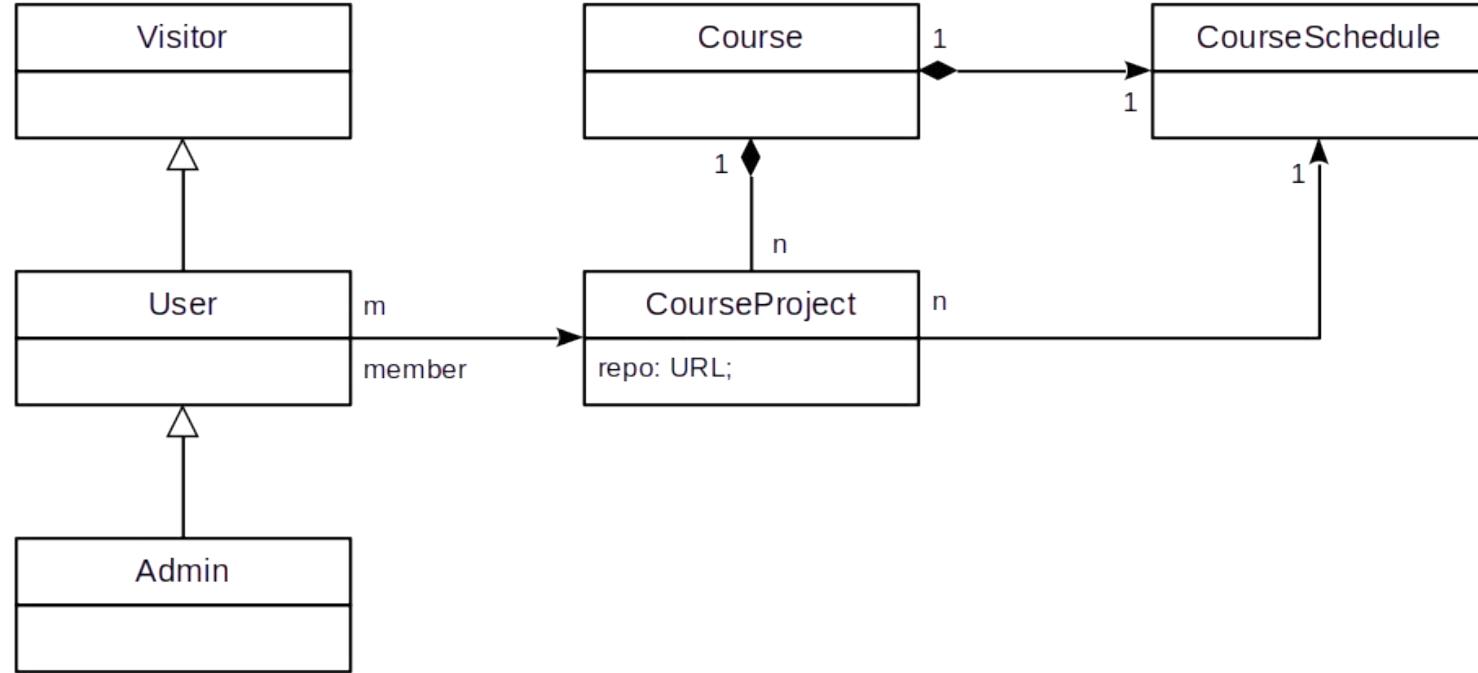
## ADAP

- One course for a given term
- Each student their own project
- Students can create their project

## AMOS

- One course for a given term
- Small number of set projects
- Students join existing project

# Class Model (Logically)



DR

6

# Original HGL Solutions



For a given project, a member

- Can enter their happiness
- Can send out stand-up emails
- Can review their coding activities

all scoped by the course schedule

# Use Your Fav IDE But Keep Cruft Out

File Edit Selection View Go Run Terminal Help ← → Q happy-go-lucky

EXPLORER

HAPPY-GO-LUCKY

- .github
- client
- docs
- node\_modules
- server
  - dist
  - node\_modules
- src
  - Config
  - Controllers
  - Exceptions
  - Managers
  - Middleware
- Models
  - Admin.ts
  - Course.ts
  - CourseProject.ts
  - CourseSchedule.ts
  - DatabaseHelpers.ts
  - GitHubRepoURL.ts
  - Password.ts
  - ProjectMember.ts
  - ProjectParticipation.ts
  - Semester.ts
- Term.ts
- User.ts
- Visitor.ts

scripts

Serializer

Services

tests

Utils

OUTLINE

TIMELINE

main\* ↻ ⚡ 0 △ 0

TS Term.ts

```
server > src > Models > TS Term.ts > setTermName
1 import { Reader } from "../Serializer/Reader";
2 import { Serializable } from "../Serializer/Serializable";
3 import { Writer } from "../Serializer/Writer";
4 import { Course } from "./Course";
5
6 export class Term implements Serializable {
7     protected id: number;
8     protected termName: string | null = null;
9     protected displayName: string | null = null;
10    protected courses: Course[] = [] // 1:N relationship
11
12    constructor(id: number) {
13        this.id = id;
14    }
15
16    async readFrom(reader: Reader): Promise<void> {
17        this.id = reader.readNumber("id") as number;
18        this.termName = reader.readString("termName");
19        this.displayName = reader.readString("displayName");
20        this.courses = (await reader.readObjects("termId", "courses")) as Course[];
21    }
22
23    writeTo(writer: Writer): void {
24        writer.writeNumber("id", this.id);
25        writer.writeString("termName", this.termName);
26        writer.writeString("displayName", this.displayName);
27    }
28
29    // Getters
30    public getId(): number {
31        return this.id;
32    }
33
34    public getTermName(): string | null {
35        return this.termName;
36    }
37
38    public getDisplayName(): string | null {
39        return this.displayName;
40    }

```

Philip Heltweg (2 months ago) Ln 47, Col 16 Spaces: 2 UTF-8 LF {} TypeScript

# SQLite Database Browser (myDatabase.db)

The screenshot shows the DB Browser for SQLite interface with the database file `myDatabase.db` open. The left pane displays the database structure, including tables, indices, and triggers. The right pane shows the content of a selected database cell.

**Database Structure:**

- Tables (9):**
  - courses:** CREATE TABLE courses (id INTEGER PRIMARY KEY AUTOINCREMENT, courseName TEXT UNIQUE, termId INTEGER);
    - id:** INTEGER "id" INTEGER
    - courseName:** TEXT "courseName" TEXT UNIQUE
    - termId:** INTEGER "termId" INTEGER NOT NULL
  - happiness:** CREATE TABLE happiness (id INTEGER PRIMARY KEY AUTOINCREMENT, projectId INTEGER, userId INTEGER, happiness INTEGER);
    - id:** INTEGER "id" INTEGER
    - projectId:** INTEGER "projectId" INTEGER
    - userId:** INTEGER "userId" INTEGER
    - happiness:** INTEGER "happiness" INTEGER
    - submissionDateId:** INTEGER "submissionDateId" INTEGER
    - timestamp:** DATETIME "timestamp" DATETIME DEFAULT CURRENT\_TIMESTAMP
  - projects:** CREATE TABLE projects (id INTEGER PRIMARY KEY AUTOINCREMENT, projectName TEXT UNIQUE, courseId INTEGER);
    - id:** INTEGER "id" INTEGER
    - projectName:** TEXT "projectName" TEXT UNIQUE
    - courseId:** INTEGER "courseId" INTEGER
  - schedules:** CREATE TABLE schedules (id INTEGER PRIMARY KEY, startDate Integer, endDate Integer);
    - CREATE TABLE sqlite\_sequence(name,seq):** CREATE TABLE sqlite\_sequence(name,seq)
  - submissions:** CREATE TABLE submissions (id INTEGER PRIMARY KEY, scheduleId INTEGER, submissionDate INTEGER, FOREIGN KEY(scheduleId) REFERENCES schedules(id));
    - CREATE TABLE terms (id INTEGER PRIMARY KEY AUTOINCREMENT, termName TEXT UNIQUE, displayName TEXT):** CREATE TABLE terms (id INTEGER PRIMARY KEY AUTOINCREMENT, termName TEXT UNIQUE, displayName TEXT)
    - CREATE TABLE user\_projects (userId INTEGER, projectId INTEGER, role TEXT, url TEXT, PRIMARY KEY (userId, projectId)):** CREATE TABLE user\_projects (userId INTEGER, projectId INTEGER, role TEXT, url TEXT, PRIMARY KEY (userId, projectId))
    - CREATE TABLE users (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, githubUsername TEXT, email TEXT, password TEXT):** CREATE TABLE users (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, githubUsername TEXT, email TEXT, password TEXT)
  - Indices (0):** Indices
  - Views (0):** Views
- Triggers (2):**
  - submissions\_insert\_trigger:** CREATE TRIGGER submissions\_insert\_trigger BEFORE INSERT ON submissions FOR EACH ROW BEGIN SELECT RA...
  - submissions\_update\_trigger:** CREATE TRIGGER submissions\_update\_trigger BEFORE UPDATE ON submissions FOR EACH ROW BEGIN SELECT...

**Right Panel (Edit Database Cell):**

- Mode:** Text
- Content:** 1
- Type of data currently in cell:** Text
- Size of data currently in table:** 1
- Identity:** Select an identity to connect
- DBHub.io Local Current Database:** Local
- Table View:** Name, Last modified, Size

# Optional Homework



See this table (same as project, but use table)

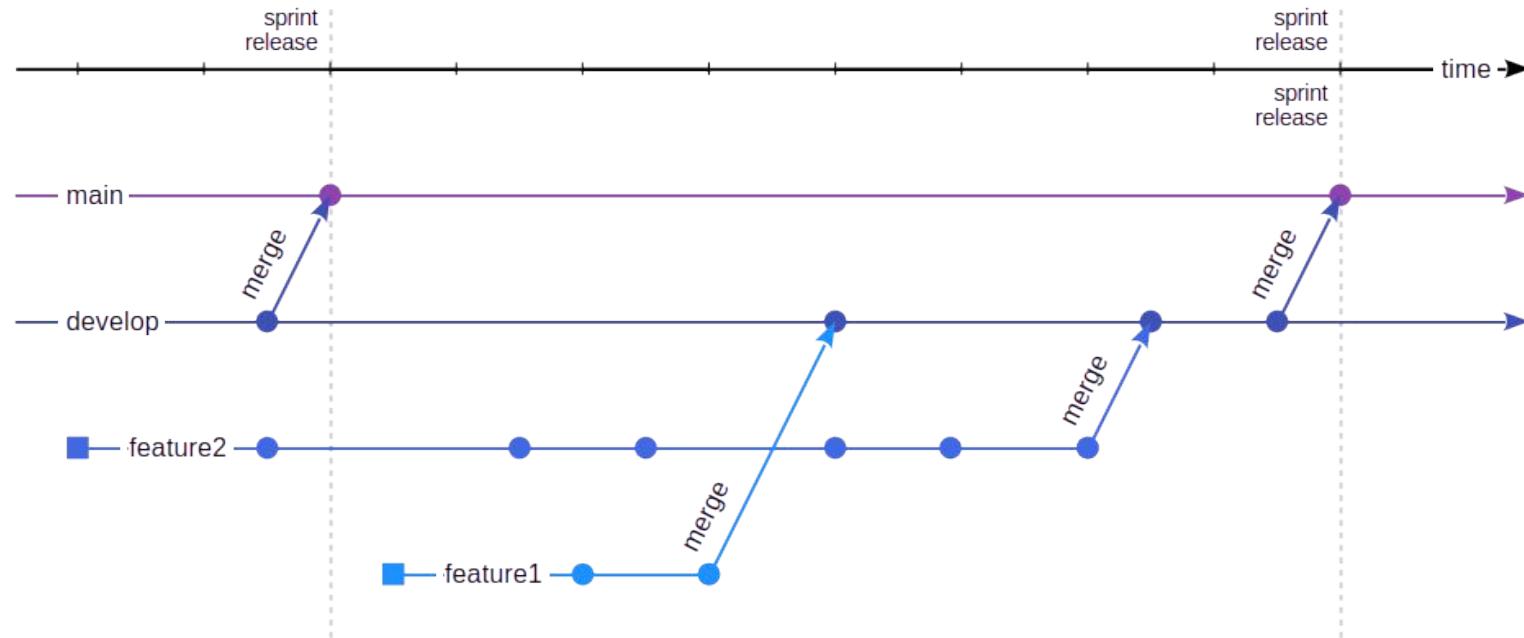
You will not be penalized if you don't participate

You'll get extra credit if you participate

Extra credit might lift your grade if you are at a grade boundary

View this experience as an idea of your first industry job

# Working With Branches



DR

11

# Homework Work and Git Flow



Fork and work from your own repository

Pick and mark item to work on (spreadsheet)

Open pull request and explain what you are going to do

Semantically chunk your work, amend your pull request

Your work may or may not be integrated

# Thank you! Any questions?



[dirk.riehle@fau.de](mailto:dirk.riehle@fau.de) – <https://oss.cs.fau.de>

[dirk@riehle.org](mailto:dirk@riehle.org) – <https://dirkriehle.com> – [@dirkriehle](#)

# Legal Notices



## License

- Licensed under the [CC BY 4.0 International](https://creativecommons.org/licenses/by/4.0/) license

## Copyright

- © 2012-2026 Dirk Riehle, some rights reserved