

# 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)

A course can have one or more projects associated with it

A course has an associated schedule (homework delivery dates)

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

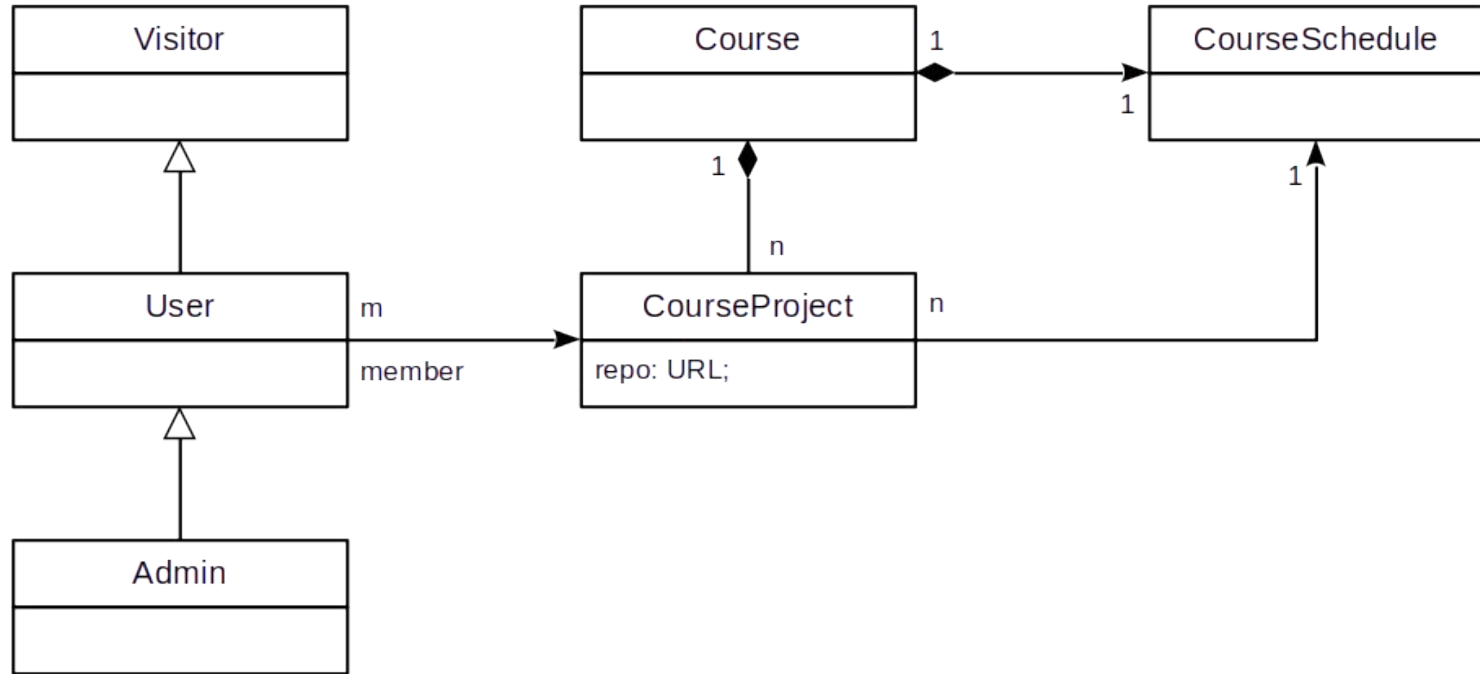
## ADAP

- 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

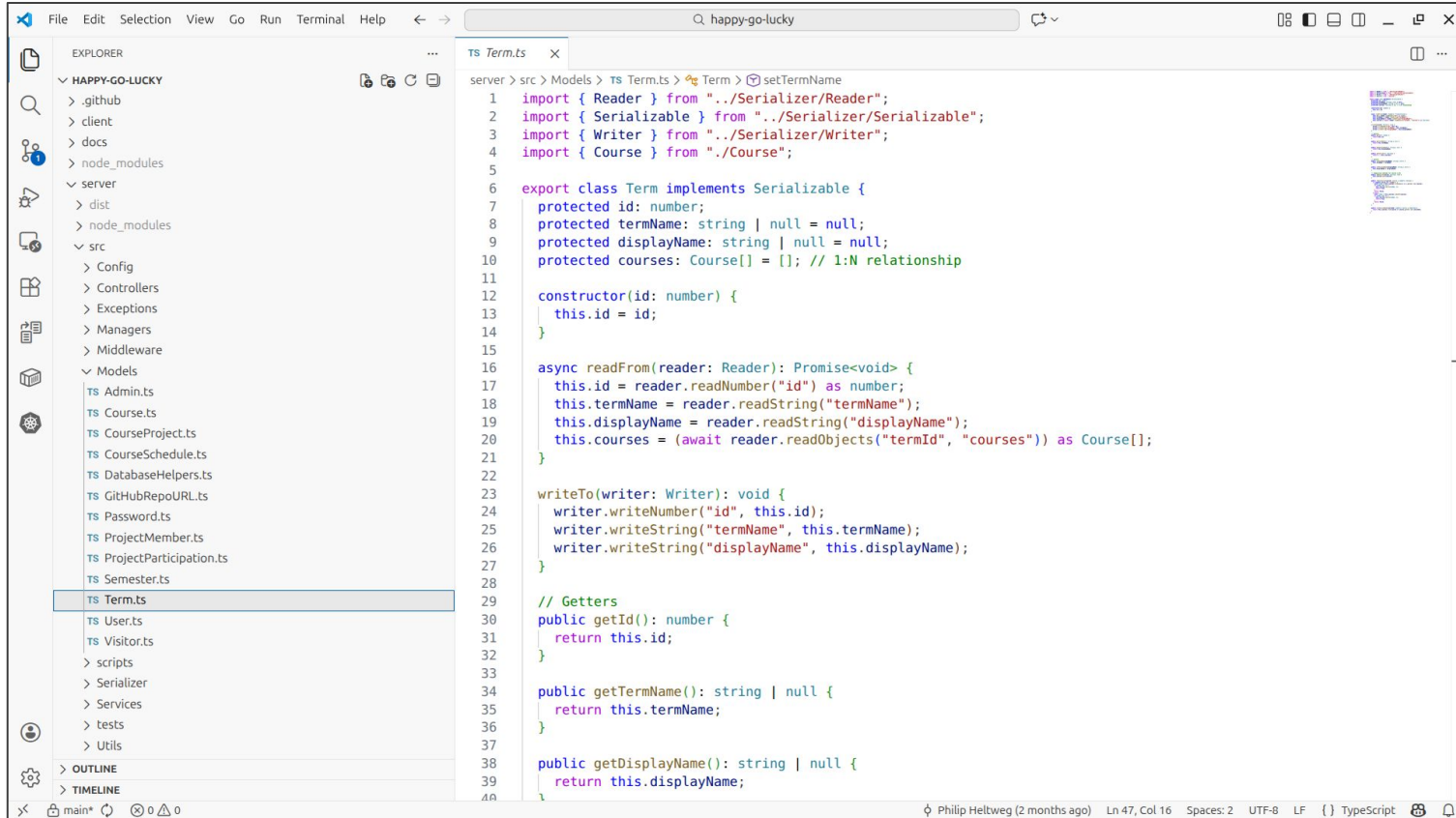
# Original HGL Solutions

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





# SQLite Database Browser (myDatabase.db)

The screenshot displays the SQLite Database Browser interface. The main window shows the 'Database Structure' tab, listing various database objects and their schemas.

Name	Type	Schema
<b>Tables (9)</b>		
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, submissionDateId INTEGER, timestamp DATETIME )
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 )
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
submissions		CREATE TABLE submissions ( id INTEGER PRIMARY KEY, scheduleId INTEGER, submissionDate INTEGER, FOREIGN KEY (scheduleId) REFERENCES schedules (id) )
terms		CREATE TABLE terms ( id INTEGER PRIMARY KEY AUTOINCREMENT, termName TEXT UNIQUE, displayName TEXT )
user_projects		CREATE TABLE user_projects ( userId INTEGER, projectId INTEGER, role TEXT, url TEXT, PRIMARY KEY (userId, projectId) )
users		CREATE TABLE users ( id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, githubUsername TEXT, email TEXT )
<b>Indices (0)</b>		
<b>Views (0)</b>		
<b>Triggers (2)</b>		
submissions_insert_trigger		CREATE TRIGGER submissions_insert_trigger BEFORE INSERT ON submissions FOR EACH ROW BEGIN SELECT RAISE(ABORT, 'insertion not allowed') WHERE ROWID != 0;
submissions_update_trigger		CREATE TRIGGER submissions_update_trigger BEFORE UPDATE ON submissions FOR EACH ROW BEGIN SELECT RAISE(ABORT, 'update not allowed') WHERE ROWID != 0;

The right-hand pane shows the 'Edit Database Cell' dialog, currently in 'Text' mode. It displays the 'Type of data currently in cell' and 'Size of data currently in table'. Below this, the 'Remote' section shows the 'Identity' dropdown set to 'Select an identity to connect'. The 'Local' tab is active, showing a table with columns 'Name', 'Last modified', and 'Size'.

At the bottom of the window, there are tabs for 'SQL Log', 'Plot', 'DB Schema', and 'Remote'. The status bar indicates 'UTF-8' encoding.

# Optional Homework

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

<https://docs.google.com/spreadsheets/d/1Td5wvsGZAJnNZjEvm8cfsthcylOhCpth3aT8vSCUZk0/edit?usp=sharing>

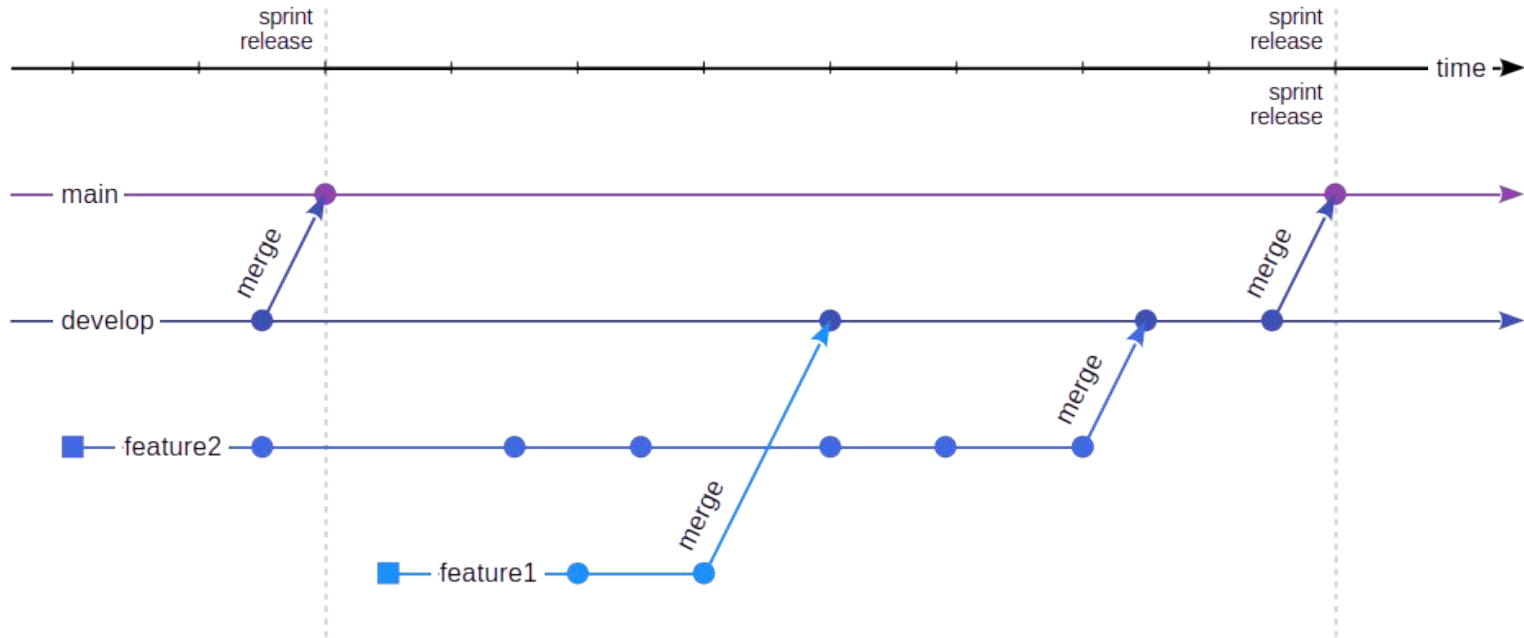
You will not be penalized if you don't participate

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



# Homework Work and Git Flow

1. Fork and work from your own repository
2. Pick and mark item to work on (spreadsheet)
3. Create a branch and a WIP pull request where you explain what you are going to do (to get feedback early on, discuss design alternatives, ...)
  - a. Source branch: feature branch on your repo
  - b. Target branch: **develop-ADAP-12** on **riehlegroup/happy-go-lucky**
  - c. Tag **@georg-schwarz** for feedback
4. Semantically chunk work into commits
5. Make sure the checks (GitHub Actions) run through successfully
6. When done, remove WIP status
7. Wait and incorporate code review feedback (amend pull request)
8. 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