Team Manager:

- Omri Shmuel

Team Members:

Full Name	Id	Phone	Email
1. Omri Shmuel	318688652	0502578783	Omri164@outlook.com
2. Astewuol Tsagaow	321072670	0532776230	Astul1996@gmail.com
3. Omer Tzaadi	207450628	0548815358	Omerzadi2014@gmail.com

Video link

App.js

```
const createError = require('http-errors');
const mongoose = require('mongoose');
mongoose.Promise = global.Promise;
const express = require('express');
const path = require('path');
const cookieParser = require('cookie-parser');
const logger = require('morgan');

const indexRouter = require('./routes/index');
const usersRouter = require('./routes/users');
const apiRouter = require('./routes/api');

const connectDB =require('./models/database');

const app = express();

// connect to mongodb
connectDB()
//mongoose.Promise = global.Promise;

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/users', usersRouter);
app.use('/api', apiRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

db.js

```
const mongoose = require('mongoose');

/**
 * @desciption MongoDB connection utility
 * @modlue database/connection
 */

const uri = "mongodb+srv://OmriShmuel:Omr!3147@cluster0.mrr1c.mongodb.net/CostManager?retryWrites=true&w=majori

/**
 * @func connectDB
 * @desc Attemps to connect to MongoDB using Mongoose
 */
const connectDB = async () => {
    try {
        await mongoose.connect(uri, {
            useNewUrlParser: true,
            useUnifiedTopology: true,
        });
        console.log('MongoDB conected');
    } catch (error) {
        console.error('Databse conection error:', error);
        process.exit(1); // Exit process with failure
    }
};

module.exports = connectDB;
```

users.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

/**
 * @desciption Defines the User schema for MongoDB
 * @modlue models/users
 */

const userSchema = new Schema({
    id: { type: String, required: true }, // Unique user iD
    first_name: { type: String, required: true }, // User's frist name
    last_name: { type: String, required: true }, // User's last name
    birthday: { type: Date, required: false }, // Date of birht
    marital_status: { type: String, required: true }, // Marrital status (single, maried, etc.)
    total_cost: { type: Number, default: 0 }
});

/**
 * @desciption Mongoose model for the Users colecttion
 */
const Users = mongoose.model('users', userSchema);

module.exports = Users;
```

costs.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

/**
 * @desciption Defines the Cost schema for MongoDB
 * @modlue models/costs
 */

const costSchema = new Schema({
    description: { type: String, required: true }, // Short desciption of the cost
    sum: { type: Number, required: true }, // The amount spent
    category: { type: String }, // Category of expense (e.g., food, education, etc.)
    userid: { type: String, required: true }, // ID of the user associated with the cost
    date: { type: Date, default: Date.now } // Date of the expense (defaults to now)
});

/**
 * @desciption Mongoose model for the Costs collection
 */
const Costs = mongoose.model('costs', costSchema);

module.exports = Costs;
```

reports.js

```
const mongoose = require('mongoose');

const ReportSchema = new mongoose.Schema({
    userid: { type: String, required: true },
    year: { type: Number, required: true },
    month: { type: Number, required: true },
    costs: { type: Object, required: true } // Stores categorized cost details
}, { timestamps: true });

module.exports = mongoose.model('Report', ReportSchema);
```

api.js

```
const express = require('express');
const router = express.Router();
const Cost = require('../models/costs');
const User = require('../models/users');
const Report = require('../models/reports');
/**
 * @route POST /api/add
 * @desc Adds a new cost entry and deletes the existing report for that month/year if it exists
 * @access Public
 */
router.post('/add', async (req, res) => {
    try {
        const { userid, category, description, sum, date } = req.body;

        if (!userid || !description || !sum) {
            return res.status(400).json({ error: 'Missing required fields: userid, description, sum' });
        }

        if (!['food', 'health', 'housing', 'sport', 'education'].includes(category)) {
            return res.status(400).json({ error: 'Category not supported. Please choose from: food, health, hou
        }

        const costDate = date ? new Date(date) : new Date();
        const cost = new Cost({ userid, category, description, sum, date: costDate });
        const savedCost = await cost.save();

        // Delete existing report for this user, month, and year (invalidate cache)
        const year = costDate.getFullYear();
        const month = costDate.getMonth() + 1; // Get correct month

        await Report.findOneAndDelete({ userid, year, month });

        // Update total cost for the user
        await User.findOneAndUpdate(
            { id: userid },
            { $inc: { total_cost: sum } },
            { new: true }
        );

        res.status(201).json(savedCost);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});


/**
 * @route GET /api/report
 * @desc Fetches monthly report for a specific user, returns from cache if available
 * @access Public
 */
router.get('/report', async (req, res) => {
    const { id: userid, year, month } = req.query;

    if (!userid || !year || !month) {
```

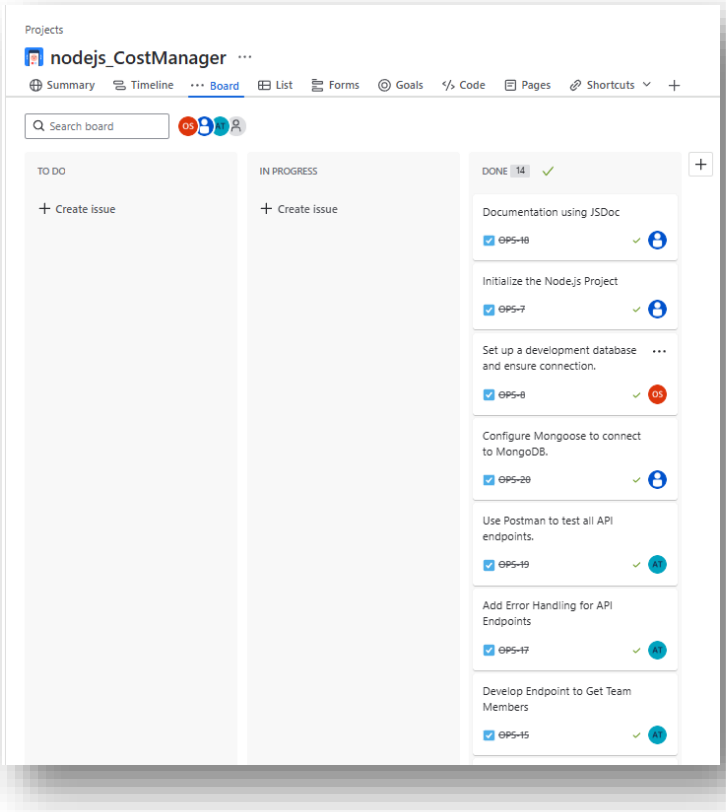
api.js

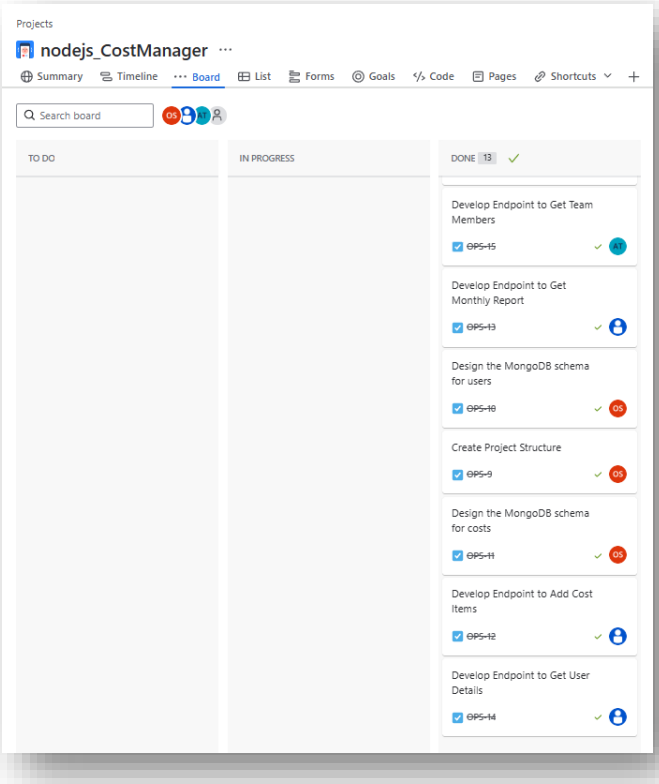
```
            return res.status(400).json({ error: 'Missing required query parameters: id, year, month' });
        }

        try {
            // Check if a report already exists in the reports collection
            const existingReport = await Report.findOne({ userid, year, month });

            if (existingReport) {
                return res.json(existingReport); // Return cached report if found
            }

            // If no existing report, generate a new one
            const startOfMonth = new Date(year, month - 1, 1);
            const endOfMonth = new Date(year, month, 0);

            const costs = await Cost.find({
                userid: userid,
                date: { $gte: startOfMonth, $lte: endOfMonth }
            });

            if (costs.length === 0) {
                return res.status(404).json({ error: 'No costs found for this user in the specified month and year'
            }

            // Categorize costs
            const categories = {
                food: [],
                education: [],
                health: [],
                housing: [],
                sport: []
            };

            costs.forEach(cost => {
                const { category, sum, description, date } = cost;
                const day = new Date(date).getDate();
                if (categories[category]) {
                    categories[category].push({ sum, description, day });
                }
            });

            const reportData = { userid, year, month, costs: categories };

            // Save report to the database
            const newReport = new Report(reportData);
            await newReport.save();

            res.json(reportData);
        } catch (error) {
            res.status(500).json({ error: error.message });
        }
    });

    /**
     * @route GET /api/users/:id
     * @desc Retrieves user details and their total spending
     * @access Public
     */
    router.get('/users/:id', async (req, res) => {
```

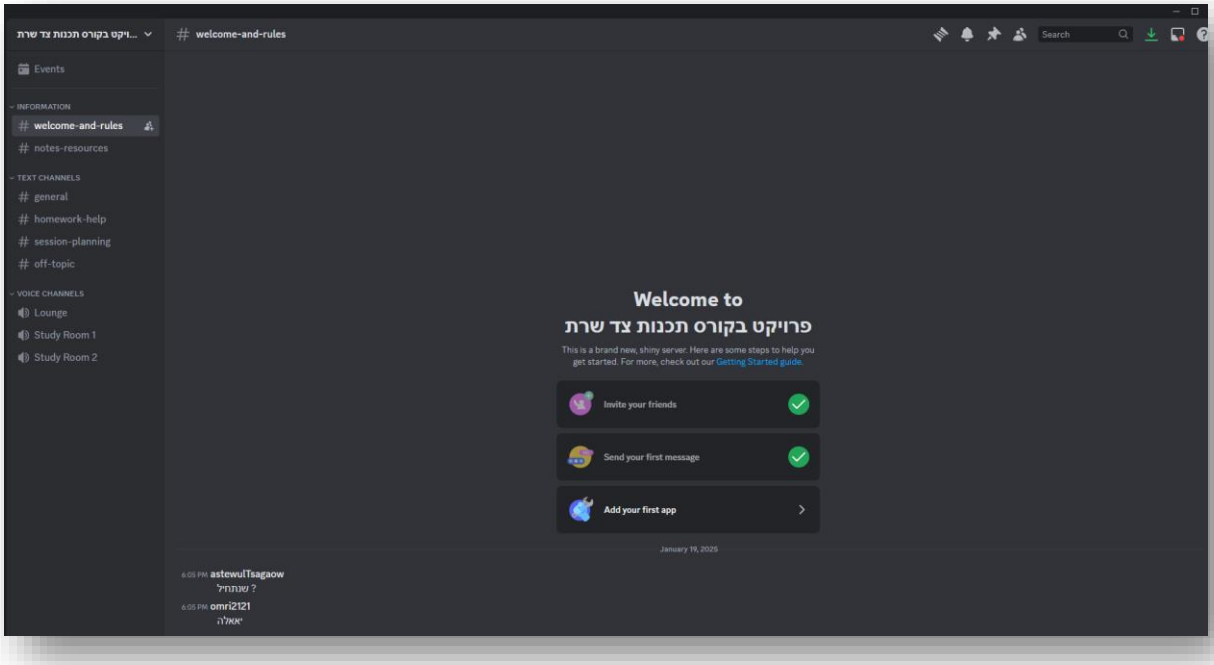
api.js

```
    const userId = req.params.id;

    try {
        const user = await User.findOne({ id: userId });
        if (!user) {
            return res.status(404).json({ error: 'User not found' });
        }

        // Calculate total costs


        res.json({
            first_name: user.first_name,
            last_name: user.last_name,
            id: user.id,
            marital_status: user.marital_status,
            total: user.total_cost,
        });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

/**
 * @route GET /api/about
 * @desc Provides information about the development team
 * @access Public
 */
router.get('/about', async (req, res) => {
    const team = [
        { first_name: 'Omri', last_name: 'Shmuel'},
        { first_name: 'Omer', last_name: 'Tzaadi' },
        { first_name: 'Astewuol', last_name: 'Tsagaow' },
    ];

    res.json(team);
});

module.exports = router;
```

Snapshots of Jira, and Slack Alternatives (Discord)






פרויקט בקורס תכנות צד שרת...
welcome-and-rules
Search
Events
INFORMATION
welcome-and-rules
notes-resources
TEXT CHANNELS
general
homework-help
session-planning
off-topic
VOICE CHANNELS
Lounge
Study Room 1
Study Room 2
Welcome to
פרויקט בקורס תכנות צד שרת
This is a brand new, shiny server. Here are some steps to help you get started. For more, check out our Getting Started guide.
Invite your friends
Send your first message
Add your first app
January 19, 2025
6:05 PM astewulTsagaow
שתתחיל ?
6:05 PM omri2121
יאללה